NSRL Project Report

Yang Li
Department of Computer Science
Saarland University
Saarbrücken, 66123
yali00004@stud.uni-saarland.de

Abstract

The paper **Reward Machines: Exploiting Reward Function Structure in Reinforcement Learning (Paper 8)** proposes the concept of reward machine, which can be seen as a finite state machine that supports to output the reward functions by taking abstracted descriptions of the environment as input. Three methodologies (automated reward shaping, HRL: task decomposition, counterfactual reasoning by experience generation) are proposed to exploit this structure, in order to program the reward function and make it visible to the agents then learn the optimal policies in a more sample efficient way. They evaluate these methods on tabular RL (Q-learning) and DRL (DDQN, DDPG) with different kinds of environments (discrete, continuous state, continuous action). The results show that these strategies do have better sample efficiency and could achieve better policies sometimes. However, the Reward Machine based on HRL (HRM) can only coverage to sub-optimal policies and the automated Reward Shaping (RS) would decrease the speed of learning for this setting. In addition, the HRM needs to use prior knowledge (not learning the self-loop and bad terminations) to improve the performance. This project will discuss and partly solve above three issues.

1 Paper Discussion

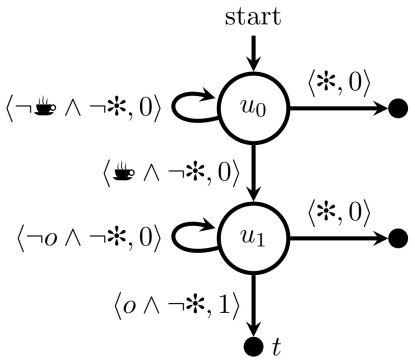

Figure 1: Simple reward machine

Figure 1 is a simple reward machine, and for more specific descriptions please refer to the original paper. An agent that has access to the specification of the reward function might be able to use such information to learn optimal policies faster. When an agent acts in the environment, moving from state to state, it also moves from state to state within a reward machine as determined by high-level

Preprint. Under review.

events detected within the environment. After every transition, the reward machine outputs the reward function (which is an integer in simple reward machine setting) the agent should use at that time.

To use this extra information to speed up learning, they also propose a collection of methods that can exploit a reward machine's internal structure to improve sample efficiency. Such as using counterfactual reasoning to generate synthetic experiences in conjunction with off-policy learning (CRM); use HRL for decomposing the problem into sub-problems that are simpler to solve (HRM); Automated Reward Shaping (RS) to provide some intermediate rewards as the agent gets closer to completing the task.

2 Issues

Although the Reward Machine would improve the performance of agent in different degrees, there are still some issues waiting to be fixed. This project means to discuss and partly solve following ones:

- They need prior knowledge about the environment to prune useless options (self-loops, bad terminal states) when using HRM, but this kind of knowledge might not be always available.
- Although HRM can be very effective at quickly learning good policies since it would learn policies for all of the options simultaneously, it might converge to sub-optimal solutions (the learned option policies will always try to transition as quickly as possible without considering how that will affect performance after the transition occurs).
- Automated RS will increase the speed of convergence in CRM but will decrease the performance of HRM.

3 Work Completed

3.1 Pre-processing and Baseline

The code of this paper has been released for further research, which is compatible with the OpenAI Gym API. After being familiar with the code and testing that it could run smoothly on my laptop, I forked and cleaned the repository by removing the auxiliary parts which are not related to the project. Now the code for this project could be found here. [1]

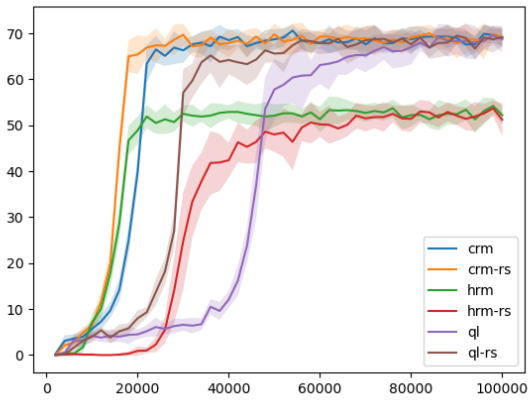

Figure 2: Baseline

The Office Task 4 environment (deliver a coffee and the mail to the office without breaking any decoration) is chosen as the baseline, with all three methods proposed in the paper (CRM, HRM, RS) on tabular Q-learning algorithm. It needs 30 minutes in total to train all settings for 10 times

[1] https://github.com/Leon950101/reward_machines.git

each. Since similar results are reported on all environments and algorithms, thus the discussion and improvement on this baseline could possibly be extended to others.

In Figure 2 we can note that CRM outperforms the other methods, and HRM learns faster than CRM (without reward shaping), but is overtaken later since it converges to a sub-optimal policy. The QL as a baseline means including the states of reward machine into policy learning. It is slower than CRM but also achieves comparable accumulative rewards at last. The automated RS helps in both CRM and QL but not in HRM.

3.2 Prior knowledge for HRM

In their experiments they do not learn options for the self-loops, since no optimal high-level policy would need to self-loop in their domains. They also do not learn options that lead to "bad" terminal states, such as breaking decorations. This project discusses the influence of self-loops and bad terminations. Figure 3 shows the states for reward machine of this task. There should have lines from 0,2,3,4 to 5 but here these lines are dropped for a simpler painting.

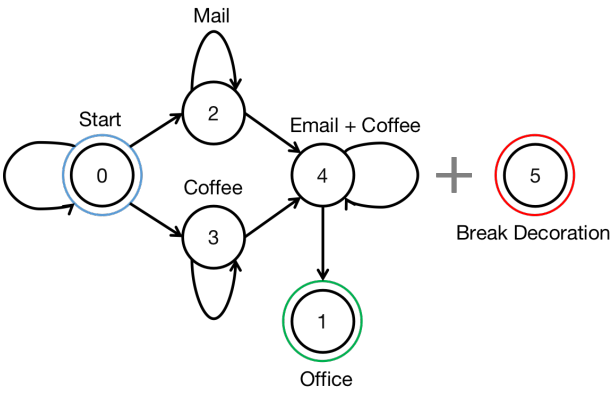

Figure 3: States for office gridworld

Figure 4 shows the experimental results. We can see that plus self-loops (sl) as options will decrease the performance of HRM a lot, with bad terminations (bt) will destroy the agent thoroughly. The agent can learn nothing with bad terminal states as options, unless we use reward hacking which gives -0.1 reward for bad terminations and +0.5 reward for one sub-goal achieved (hrm-bt-rh).

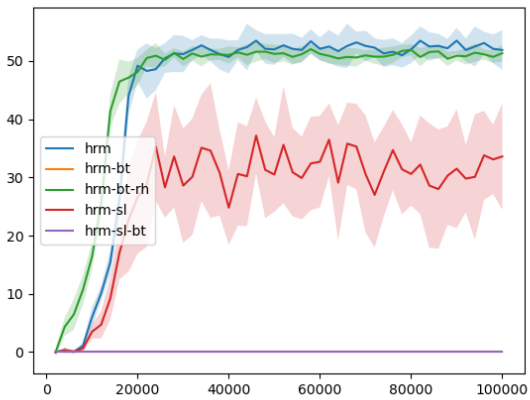

Figure 4: Prior knowledge for HRM

3.3 HRM sub-optimal convergence issue

An example of this behaviour is shown in Figure 5. The task consists of delivering a coffee to the office (Task 4 which this project uses also needs to collect the email before going to office, here just a simpler version for explanation). As such, the optimal high-level policy will correctly learn to go for the coffee and then go to the office. However, the optimal option policy for getting the coffee will move to the closest coffee station (following the sub-optimal red path) because (i) that option gets a large reward when it reaches the coffee, and (ii) optimal policies will always prefer to collect such a reward as soon as possible. As a result, HRM will converge to a sub-optimal policy.

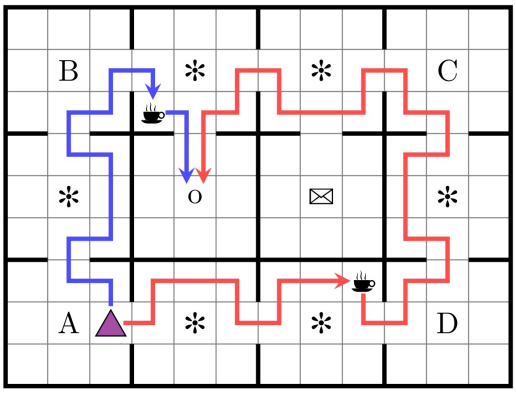

Figure 5: The office gridworld

For an option which has multiple goals this issue might usually happen. In order to solve this, the agent actually could learn every goal for each option and then decide which one to keep based on overall rewards. When the option learned how to achieve the best goal, then it should consider if there are still other sub-optimal goals. If so, learn that strategy also (If the situation of goals are known, these policies could be learned simultaneously). The high-level policy then will learn how to choose the best combination of options for higher accumulative rewards. It could be called as Far-sighted HRM and the results in Figure 6 shows that the Far-sighted HRM (hrm-fs) could achieve as good as performance with even CRM.

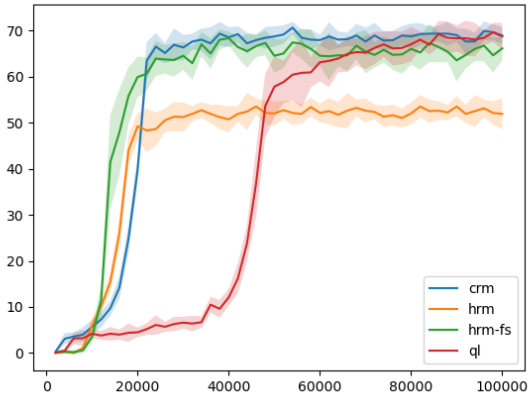

Figure 6: Far-sighted HRM

The training expense for multiple goals will depend on the number of goals and the degree of parallelization. The implementation for this idea has not been done yet and I would try to finish it in the future (for now there is just a cheating way to get the result).

3.4 RS influence for HRM

Automated RS will decrease the performance of HRM, since the intermediate reward given by potential function will confuse the option learning. The option has its own reward r^+ and r^-, but the intermediate reward will change it thus decrease the convergence speed of options (see the equation below). Thus, it might be a better idea to not use the Automated RS for HRM in practical.

$$r_{u,u_t}(s,a,s') = \begin{cases} \delta_r(u)(s,a,s') + r^+ & \text{if } u_t \neq u \text{ and } u_t = \delta_u(u, L(s,a,s')) \\ \delta_r(u)(s,a,s') + r^- & \text{if } u_t \neq u \text{ and } u_t \neq \delta_u(u, L(s,a,s')) \\ \delta_r(u)(s,a,s') & \text{otherwise} \end{cases}$$

4 Future work

The Far-sighted HRM could be extended to other environments in the future. However, since the current implementation of this is quite basic, the code should be modified first. In addition, there are other issues in this paper could be discussed, such as labelling function dependence problem.