Free-time Exploration in Reinforcement Learning

Masih Hashemi
School of Computing
Queen's University
Kingston, Ontario
masih.hashemi@queensu.ca

Paul Wilson
School of Computing
Queen's University
Kingston, Ontario
1pfrw@queensu.ca

***Abstract*—Exploration is an integral component of solving Reinforcement Learning (RL) tasks. RL agents need to recognize when to explore the environment and when to exploit its rewards. Typical exploration strategies, such as those using curiosity or novelty rewards to motivate exploration, have an associated risk of missing out on rewards. We propose a *free* exploration strategy using a surrogate problem: learning *free-time*. In this context, the agent learns the value of state-action pairs *and* the amount of free-time associated to them. Free-time is defined as time for which multiple actions could lead to the same amount of future discounted reward. In this paper we demonstrate how exploring during free-time can lead to improved sample efficiency, improved understanding of the environment and better performance under certain conditions.**

***Index Terms*—Reinforcement Learning, Exploration, Q-Learning**

I. Introduction

A significant focus of research in reinforcement learning is on the exploration-exploitation trade-off, aiming to answer questions related to *when*, *where*, and *how much* to explore. It is usually necessary to explore more at the beginning of learning, or in less visited territories of the environment. But when should the agent stop? Simple or naive exploration strategies which involve random exploration with certain probabilities (e.g. $\epsilon - greedy$ exploration) have been shown to provably fail in tasks complex with state spaces (1). More advanced approaches have looked at curiosity or novelty rewards of some form to encourage exploration. While these approaches are proven to be sample-efficient in theory, in practice these reward functions are engineered iteratively to incentivize a desired approach to exploration within the task (2). This fact implies that these strategies may require repeatedly employing the same RL algorithm with differing reward functions which could be sample inefficient. In this paper, we propose a "free" method of exploration: by maximizing exploratory behaviour only when it is known that exploration would not lead to a loss of future discounted reward.

A. Background and Problem Formulation

The on-line reinforcement learning problem is usually defined by an stationary Markov Decision Process (MDP) given by a tuple $< S, A, P, R, \gamma >$, where S is the state space, A is the action space, $P(s'|s,a)$ is the probability of transiting from state s to state s' when selecting action a, $R(s,a,s')$ is the expected reward r when transiting from state s to state s' using action a, and $0 \leq \gamma \leq 1$ is the discount factor. The problem is to maximize the sum of discounted future rewards $\sum_{t=0}^{T} \gamma r_t$ where T can be ∞ given $\gamma < 1$.

When the MDP is unknown, the difficulty lies in the fact that learning the optimal policy $\pi^*(s)$ (which returns an action a) or $\pi^*(s,a)$ (which returns a probability for each action a) that maximizes rewards, requires the learner to first explore and visit every state-action pair (infinitely) often. This is the exploration/exploitation problem. In model-free RL, one attempts to learn this policy without learning an estimates of P and R. In model-based RL, one attempts to build such estimate, allowing it to refine its policy off-line as it accumulates information about the model. To learn the policy, one can build an estimate of the expected future rewards $Q(s,a)$ for each state-action pair, where the relative scores for each action is used to define the policy function. Finally, an algorithm could be on-policy, meaning it learns the value function for the current policy, or off-policy, in which case, it can learn a policy (let say, the optimal one), while executing a different policy (such as an exploration policy).

B. Related Work and Proposed Method

Since a necessary condition for learning the optimal policy $\pi^*(s)$ is that we need to visit each state or state-action pair infinitely often, it is crucial for any RL agent interacting with the environment to employ *exploration* strategies to visit states and

investigate their reward output. While simple randomized exploration strategies are useful in basic tasks, their effectiveness dwindles when the state and action spaces are large or when precise sequences of actions are needed to reach states with high reward: For example it takes exponential time for an agent to reach a corner reward in the environment using a random walk (1).

More complex strategies for exploration such as count-based exploration, "Intrinsic" or "Curiosity" rewards have been studied in the recent years (3) (4), but they have associated challenges since they either require enumerating over the state-action space, or learning state-transition models. They become impractical with growing dimensionality (5). Furthermore, all the mentioned exploration strategies suffer from inherit risk of missing out on reward in favor of exploring the environment, which is why this is problem called the exploration/exploitation *trade-off*.

We propose a "free" exploration strategy: exploring during *free-time*. Free-time may be defined as time-steps in which the agent can take multiple actions which lead to a maximum amount of future discounted rewards. This means that during a period of free-time, a sub-set or potentially all actions in the action space can be chosen by the agent without loss of opportunity to obtain rewards. Consider a situation in the Atari game Pong. A player would have free-time in a given time-step if the ball is far enough from the a player's paddle so that movement in any direction would not compromise the player's ability to hit the ball with its paddle in the future. In this context, the agent learns the value of state-action pairs $Q(s,a)$ *and* the amount of free time associated to them $F(s,a)$.

We demonstrate that by training the agent to learn when it has the liberty of free-time and having the agent maximize exploration during periods free-time we obtain better sample efficiency and knowledge of environment at no cost to the performance of the agent. In fact, in certain tasks and under certain initial conditions we demonstrate that the agent which maximizes exploration during free-time significantly outperforms the baseline in terms of score obtained in the task.

II. Methodology

A. Theory of free-time

A free action is one that does not decrease the expected returns compared to an optimal action. Consider the state value function for an optimal policy π^* expressed in terms of an expectation over the next state:

$$v_{\pi^*}(s) = \mathbb{E}_{\pi^*(s)}[R_{t+1} + \gamma v_{\pi^*}(S_{t+1}) | S_t = s] \quad (1)$$

Suppose now that an action a is chosen while in state s not necessarily following π^*. We say that this action is "free" if and only if

$$v_{\pi^*}(s) \leq \mathbb{E}[R_{t+1} + \gamma v_{\pi^*}(S_{t+1}) | S_t = s, A_t = a], \quad (2)$$

Let us define $I_{\text{free}}(s,a)$ to be the indicator function that equation 2 is satisfied for the pair (s,a). Note that substituting the optimal policy π^* with a policy choosing any free action in a state s, and following π^* elsewhere, results in a policy that is still optimal. We also define the "free-time" value of a state action pair:

$$f(s,a) = I_{\text{free}}(s,a) * \quad (3)$$
$$(1 + \mathbb{E}[\max_{a'}(f(S_{t+1}, a')) | S_t = s, A_t = a]). \quad (4)$$

Our method is to learn I_{free} and f for state-action pairs through direct experience and use them to guide behavior. They can be learned and updated online using sample estimates in a temporal-difference learning fashion. For a sequence s, a, r, s' (state, action, reward, next state) observed via interaction with the environment, we obtain a local estimate:

$$I_{\text{free}}(s,a) \approx \begin{cases} 1 & \text{if } v_{\pi^*}(s) \leq r + \gamma v_{\pi^*}(s') \\ 0 & \text{otherwise} \end{cases}, \quad (5)$$

which can be used to update the free-time value function via the rule:

$$\delta_{\text{free}} = I_{\text{free}}(s,a)(1 - \max_{a'} f(s',a)) - f(s,a); \quad (6)$$

$$f(s,a) \leftarrow f(s,a) + \alpha_{\text{free}}\delta. \quad (7)$$

Making this estimate requires theoretical knowledge of the optimal state-value function $v_{\pi^*}(\cdot)$, which is not known in general. Therefore, we make use of local estimates of this state value function which obtained from a Q-table learned online as well, namely $v_{\pi^*} \approx \max_{a'} Q(s,a')$. In principle, this Q-table could be used to check equation 2 directly, removing the need to estimate it with this update rule. We discuss this apparent contradiction further in section IV-A. In any case, we observe strong experimental evidence for benefits of our approach.

B. Learning Free time

Our algorithm (algorithm 1) extends the standard Q-learning algorithm by learning the f-value of state action pairs concurrently with their q-value, utilizing F and Q tables for current approximations. Given an observation of state, action, reward, and next state (s, a, r, s'), we update the Q-table via the generic Q-learning update

$$Q(s,a) \leftarrow Q(s,a) + \alpha(r + \max_{a'} Q(s',a') - Q(s,a)), \tag{8}$$

and we update $f(s,a)$ based on the update rule derived from equations 6 and 7, with a the allowance of a small "tolerance" hyperparameter τ which allows $I(s,a) = 1$ as long as

$$\max_{a'} Q(s,a') \leq r + \gamma \max_{a'} Q(s',a') + \tau. \tag{9}$$

Compared to the strict definition in equation 5, a larger tolerance parameter allows an action to be more readily as interpreted as "free".

The most important difference of our algorithm compared to Q-learning is that the behavior policy is derived from both the F and Q values. Based on the heuristic that actions with F scores are very likely to be free in the sense of equation 2, any of these actions are considered to be good candidates to try. Therefore, an action at random from among $\mathcal{A}_f(s) = \{a | f(a,s) \geq 1\}$. Only if this set is empty to we fall back to the Q-learning and choose a greedy action according to the q-values. We believe this behavior is critical to the success of our method, as it balances increased exploration with reverting to greedy behavior when necessary. We call this policy the "free-time policy"

III. Results

In order to evaluate our free-time algorithm we performed experiments testing the performance of our methodology compared to baseline Q-learning algorithm in multiple grid-world environments. The aim of these experiments was to analyze the behaviour of the free-time agent across environments with differing level of complexity.

Specifically, these tasks were designed in manner such that sufficient exploration was needed to find rewards. In certain environments, an agent who was not exploring robustly may continuously exploit a lesser reward source when a greater one was available. Different initializations of the Q-table were used in all experiments. For each experiment, a search of the best hyperparameter τ was done and the results reported used this value. The hyperparameters were fixed with the values $\alpha = 0.01$, $\alpha_f = 0.01$, $\epsilon = 0.05$.

A. Environment 1: Windy Single Reward

To assess the feasibility of the free-time algorithm in a basic table based task, we created a windy grid-world environment where a reward is fixed at the top center position of the grid. The agent will start randomly with uniform probability at any of the positions on the bottom row of the grid. The agent can then either choose to move left, right or stay in its current position but will be moved up a single state at each time-step no matter its choice of action. The agent will receive a reward of 1 if it reaches the top center grid. If at any point the agent takes and action that removes it from the boundary of the grid, the episode will end with a reward of zero. It is clear that this task is one where the agent will have a "cone of free-time" where as it's moving up toward the reward due to the wind, more than one possible action can lead to a path to the reward as long as the agent is within this cone. The aim of this experiment was to show firstly that free-time can be learned by an agent and secondly to assess the impact of maximising exploration during free time. The experiment was run under three different initial conditions of the Q table: Q table initialized with all state-action values 1 (Optimistic), Q table initialized with all state-action values 0 (Pessimistic), Q table with state-action values distributed uniformly between 0 and 1 (Random).

TABLE I
AVERAGE CUMULATIVE REWARDS AFTER $100,000$ TIMESTEPS, ENVIRONMENT 1

	Random	Pessimistic	Optimistic
free-time (ours)	**3052.0**	4672.0	**2546.9**
Vanilla Q-learning	2824.4	**4912.4**	2545.7

When compared to vanilla Q-learning as a baseline, the free-time algorithm shows improved and more consistent performance (cumulative sum of rewards over fixed number of time-steps) when the Q-table was initialized randomly. Furthermore, under all Q-table initial conditions of random, pessimistic, and optimistic, the free-time algorithm was able to provide a more thorough and accurate representation of the state-action values at little or minimal cost to performance. This better understanding of the state space is demonstrated in figure 1

B. Environment 2: Multiple Obtainable Rewards

In order to test the robustness of this approach in dealing with more complicated situations where free-

Algorithm 1 free-time Q-Learning

Set Algorithm parameters: $\alpha, \alpha_{\text{free}} \in (0, 1]$, small $\epsilon > 0$, small tolerance $\tau \geq 0$
Initialize $Q(s, a)$, $F(s, a)$ for all $s \in \mathcal{S}$, $a \in \mathcal{A}$ {e.g $F(s, a) = 1$}
for Each episode **do**
  Initialize starting state s_1
  for Each step 1,2...,T **do**
    if $\exists\, a \in \mathcal{A}$ such that $F(s_1, a) \geq 1$ **then**
      Choose action a randomly among $\{a \in \mathcal{A} | F(s_1, a) \geq 1\}$
    else
      Choose a from s_1 using policy derived from Q (e.g. ϵ-greedy)
    end if
    Take action a, receive reward r, and new state s_2
    $Q(s_1, a) \leftarrow Q(s_1, a) + \alpha[r + \gamma \max_{a' \in \mathcal{A}} Q(s_2, a') - Q(s1, a)]$
    if $\max_{a' \in \mathcal{A}} Q(s_1, a') \leq r + \gamma * \max_{a' \in \mathcal{A}} Q(s_2, a') + \tau$ **then**
      free-time target: $t_f = 1 + \max_{a'} f(s, a')$
    else
      free-time taget: $t_f = 0$
    end if
    $F(s_1, a) = F(s_1, a) + \alpha_f(t_f - F(s1, a))$
    $s_1 = s_2$
  end for
end for

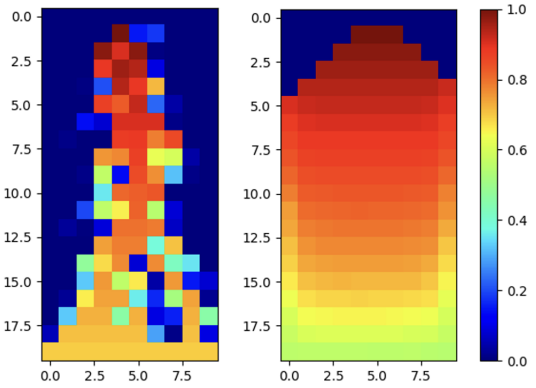

Fig. 1. Sample heatmaps representing the maximum Q-table values for environment 1. Left: results of vanilla Q-learning algorithm. Right: results of the free-time algorithm. The free-time algorithm learns a more complete and correct understanding of q-values across all states in the environment.

time is available we schemed a similar environment to Environment 1. This time two rewards are present on the grid in positions (1,6) and (3,4). This means that the rewards differ by 2 vertical and 2 horizontal positions and thus a second smaller free-time cone is present after the agent obtains the first reward. The results of this experiment showed that the agent was able to learn both the free-time cone pertaining to the initial reward and a second cone that exists between the two rewards. Furthermore all of the benefits of a more accurate representation of the state-action values observed in experiment 1 were also transferred to this task. Most notably however, in this task we observed that at under random and pessimistic initialization of the Q table, the vanilla Q-Learning agent was susceptible to missing one of the rewards in the preliminary episodes. The free-time algorithm ensured that under all initial conditions the agent found both rewards quickly which further exhibited the benefits of sample efficiency offered by this approach. This can be observed by the improved performance of the free-time agent in Table II.

TABLE II
AVERAGE CUMULATIVE REWARDS AFTER $100,000$ TIMESTEPS, ENVIRONMENT 2

	Random	Pessimistic	Optimistic
free-time (ours)	**6421.8**	**8950.2**	8808.7
Vanilla Q-learning	3765.6	8710.1	**8842.3**

C. Environment 3: Corner Rewards

This experiment is meant to demonstrate the ability of this approach to uncover new reward signals quickly. In this version of the windy grid-world, two rewards of differing values are placed at each top corner of the grid. In certain trials of this experiment, we observed that if the Q-learning agent were to discover and exploit the lower valued reward first, it would take many trials for it to discover the higher

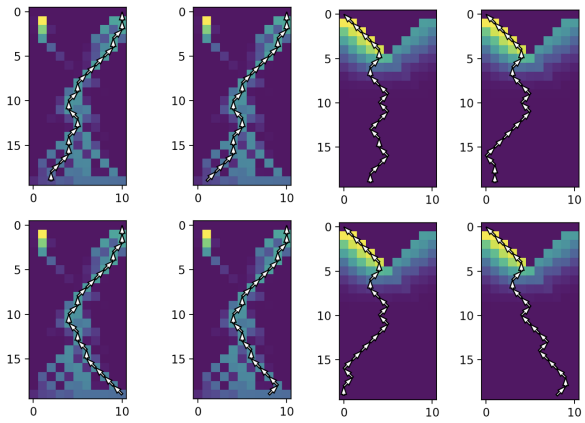

Fig. 2. Sample trajectories over heatmaps representing the maximum Q-table values for environment 3. Left: results of vanilla Q-learning algorithm. Right: results of the free-time algorithm. The reward at the top left is more valuable. The free-time algorithm discovers the more valuable reward more quickly and overcomes any initial confirmation bias causing it to mistakenly prefer the top right reward.

valued reward at the opposite corner. This issue was once again specifically prevalent under random and pessimistic initial conditions. In this task we showed that when maximizing exploration by exploiting free-time, the agent was able to consistently discover the highest valued reward, even if the lower valued reward was the first to be discovered and exploited.

TABLE III
AVERAGE CUMULATIVE REWARDS AFTER $100,000$ TIMESTEPS, ENVIRONMENT 3

	Random	Pessimistic	Optimistic
free-time (ours)	**18992.3**	**27625.9**	**38642.3**
Vanilla Q-learning	14282.4	12364.5	38230.6

D. Environment 4: Windless World

In order to ensure that the free-time algorithm was not hindering performance, we ran both Q-learning and the free-time algorithm in a wind free environment. In this environment the agent once again starts in a random state on the bottom row of the environment. A single reward is located in the center of the world. Even though the intention of this trial was to demonstrate that the free-time algorithm does not impact performance in a windless environment, we found that the free-time algorithm resulted greatly improved cumulative rewards in under both pessimistic and random initialization of the Q-table(as seen in Table IV) all while not impacting the performance under optimistic initial conditions. An assessment of the Q-values and trajectory maps suggests that the agent (likely due to increased exploration) is more likely to find the shortest path to obtaining the reward (see Figure 3).

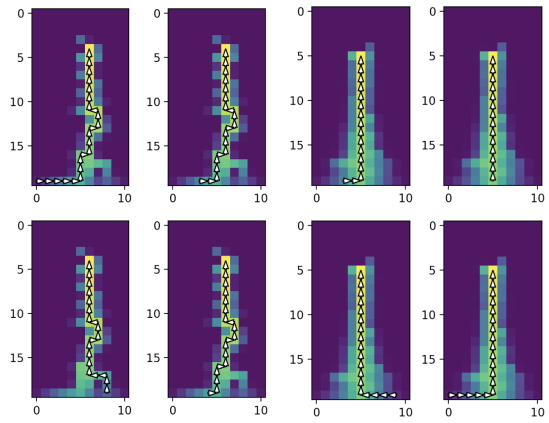

Fig. 3. Sample trajectories over heatmaps representing the maximum Q-table values for environment 4. Left: results of vanilla Q-learning algorithm. Right: results of the free-time algorithm. The free-time algorithm learn the more direct path to the reward.

TABLE IV
AVERAGE CUMULATIVE REWARDS AFTER $100,000$ TIMESTEPS, ENVIRONMENT 4

	Random	Pessimistic	Optimistic
free-time (ours)	**567.8**	**7335.7**	829.4
Vanilla Q-learning	115.9	6708.3	**829.7**

IV. DISCUSSION

A. Why This Works

As mentioned in Section II-A, equation 2 can be in theory be checked directly using the q-value function. In fact, assuming the q-value function was known, this function could be re-written as

$$\max_{a'} q(s, a') \leq q(s, a), \tag{10}$$

and we are simply asking "is action a tied for the maximum q-value?". Why, then, is there any benefit to estimating the free-time value function via the update rules defined in equations 6 and 7, rather than through direct observation of the Q-learning? Furthermore, any free action would be tied for highest q-value and therefore the behavior of the free-time policy would already be captured by a greedy policy based on q. On the surface, it would appear that we are adding a redundant mechanism to track information already being calculated via the baseline algorithm. However, we have shown experimentally that our algorithm shows significantly improved performance and very different behavior than baseline Q-learning. We speculate on a few reasons for the success of this algorithm.

Firstly, we think that the free-time policy allows us to prevent early biases in the Q-learning from becoming self-reinforcing. Early on in training, it is highly likely due random chance that a single action among the available actions in the state has higher q-value among the others even when they are theoretically of equal value. With standard Q-learning, this choice of action subject to a "confirmation"bias where it is chosen over and preference for this action increases further.

Another hypothesis is as follows: The bias in the Q-learning towards the perceived "optimal" action leads to increased experience with this action. Consequently that $\max_{a'} q(s, a')$ is a better estimate of $v_{\pi^*}(s)$ than $q(s, a)$ is of the value of a for another arbitrary action a. Therefore, utilizing equations 6 and 7 with $v_{\pi^*}(s) \approx \max_{a'} q(s, a')$ is more accurate than using equation 10 directly to determine whether or not an action was free. In this sense the free-time updates actually *exploit* the confirmation bias in the Q-learning.

B. Sensitivity to Hyperparameters

As updates to the free-time depend on estimates of the q-value function which are initially biased, it is possible for such biases to transfer to the free-time value function. We introduce the tolerance hyperparameter τ to control the extent to which actions which "nearly" satisfy equation 2 are considered to be free. We found that our algorithm was more sensitive than initially expected to this hyperparameter. In some experiments, a small change (eg. 0.001 to 0) made a large difference to the final performance. The algorithm was more sensitive to hyperparameters under random and pessimisitic Q-table initializations than optimistic Q-table initializations. Table V contains a grid search of the best values for τ for the different initializations in environment 3. There is no obvious trend for which value of τ will perform the best for a given environment and initialization. A more thorough theoretical and experimental analysis of this hyperparameter is left to future work.

TABLE V
PERFORMANCE FOR DIFFERENT VALUES OF τ IN ENVIRONMENT 3

	-0.01	-0.001	0	0.001
Random	17718.9	13085.5	18547.2	8481.6
Pessimistic	23871.5	23750.2	20382.5	21955.1
Optimistic	37473.2	38243.5	38123.0	38170.4
	0.01	0.1	0.5	Baseline
Random	13765.7	13707.3	9931.6	9698.4
Pessimistic	24422.6	19739.3	21550.5	7358.6
Optimistic	38460.6	37514.3	37915.3	38829.1

C. Future Work

Future work will primarily focus on extending the free-time learning concept to a deep reinforcement learning paradigm. The free-time algorithm shows the greatest performance benefits when used with randomly initialized Q-tables. Because of the nature of q-value approximations using deep neural networks, whose weights are randomly initialized, we expect that the free-time concept may extend well to this methodology. In particular, we propose the use of a neural network approximating the F-table, which would be trained in conjunction with a network approximation the Q-table, with an analogous extension of the update rules defined in section II-A. In addition, this methodology will be extended to more complex environments with larger state-spaces and stochastic state transition properties.

ACKNOWLEDGMENT

This work was done under the supervision of Dr. Francois Rivest.

REFERENCES

[1] L. Li, "Sample complexity bounds of exploration," in *Reinforcement Learning*. Springer, 2012, pp. 175–204.

[2] C. Jin, A. Krishnamurthy, M. Simchowitz, and T. Yu, "Reward-free exploration for reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2020, pp. 4870–4879.

[3] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," in *International conference on machine learning*. PMLR, 2017, pp. 2778–2787.

[4] G. Ostrovski, M. G. Bellemare, A. Oord, and R. Munos, "Count-based exploration with neural density models," in *International conference on machine learning*. PMLR, 2017, pp. 2721–2730.

[5] R. Ramamurthy, R. Sifa, M. Lübbering, and C. Bauckhage, "Novelty-guided reinforcement learning via encoded behaviors," in *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020, pp. 1–8.