# ReversedQ: Opportunities for Faster Q-Learning in Episodic Online Reinforcement Learning

Sofia R. Miskala-Dinç
University of Maryland
College Park, MD, United States
smiskala@umd.edu

Aviva Prins
University of Maryland
College Park, MD, United States
aviva@umd.edu

## ABSTRACT

We study model-free $Q$-learning in finite-horizon episodic Markov Decision Processes (MDPs) with stationary dynamics across episodes. We identify a central issue in nascent model-free posterior-sampling works: the reliance on delayed learning in order to prove theoretical guarantees. In particular, we identify three opportunities for faster learning – (i) value-function update order, (ii) update frequencies, and (iii) value-function initialization. Using Wang et al. [13]'s `RandomizedQ` as a basis, we illustrate these changes and their individual (as well as cumulative) impact in multiple empirical studies. We find that our combined modifications, termed `ReversedQ`, improve scaled mean cumulative reward compared to `RandomizedQ`, from 9.53% to 78.78% in the Bidirectional Diabolical Combination Lock (BDCL), and from 21.76% to 61.81% in a chain MDP.

## KEYWORDS

Q-learning, Episodic reinforcement learning, Model-free algorithms, Bayesian exploration

## 1 INTRODUCTION

Reinforcement Learning (RL) algorithms aim to maximize received reward by directing an agent to take actions that affect the environment. These algorithms are useful for solving real-world problems such as inventory control, crop-planning decision support, and robotics [5–7]. Many such settings are naturally modeled as episodic environments, where interaction proceeds for a fixed number of time steps and then resets. A core challenge in episodic RL is managing the exploration-exploitation tradeoff: to reduce uncertainty about the environment, an agent must sometimes explore actions that may be suboptimal. In order to do this, an algorithm must encourage risk-taking or exploration in some way.

Traditional frameworks predominantly rely on upper confidence bounds, which incentivize exploration via additive bonuses [2–4]. However, these bonuses often prove to be overly conservative. Conversely, posterior-sampling methods–which guide exploration through policy variance derived from a posterior distribution–frequently exhibit superior empirical performance despite a persistent gap in their theoretical characterization [9, 11]. Recent work by Wang et al. [13] made progress in bridging that gap by introducing `RandomizedQ`, which has both an established sublinear regret bound $\tilde{O}\left(\sqrt{H^5SAK}\right)$ and step-wise policy updates (in contrast to stage-based updates). As a state-of-the-art model-free posterior-sampling algorithm with both a provable regret bound and step-wise updates, `RandomizedQ` provides a natural baseline for isolating the practical costs of theoretically conservative mechanisms. Demonstrating that its practical performance can be substantially improved is therefore directly relevant to the broader goal of algorithms that are both efficient and deployable.

Although Wang et al. [13] correctly prioritize "agility" (that is, to emphasize responsiveness to new observations), their proposed algorithm overlooks several avenues for enhancing this responsiveness in practical domains. Specifically, we focus on value-function update order, update frequencies, and value-function initialization. We illustrate how these changes may be implemented, using `RandomizedQ` as a basis, in `ReversedQ` (Algorithm 1). Our aim is to foster collaboration on bridging this gap between theory and practice, and to provide intuition for those who work on more complex variants of these systems (i.e., nonstationary or multi-agent systems).

We empirically evaluate the concept of faster learning rates across Bidirectional Diabolical Combination Lock (BDCL) [1], and a chain MDP. Our evaluations show that `ReversedQ` significantly increases total reward, achieving a scaled mean cumulative reward of 78.78% ± 0.75 in BDCL and 61.81% ± 2.23 in the chain MDP, compared against the baseline's 9.53% ± 11.04 and 21.76% ± 1.11, respectively. These findings provide strong evidence that our approach offers a marked improvement over `RandomizedQ`. For practitioners and researchers designing agents for complex, finite-horizon environments, this demonstrates that overcoming the conservative mechanisms required for theoretical regret bounds can yield substantially more agile and efficient exploration in practice.

## 2 RELATED WORKS

The central problem in online RL is optimally balancing exploration and exploitation. That is, how to trade off the potential benefit of exploring unknown system dynamics versus leaning into the estimated benefit from past experience. We can divide the approaches in the literature into two groups: bonus-based optimism and Bayesian-based posterior-sampling.

Approximating unknown quantities without sufficient exploration can lead to premature convergence to suboptimal policies. To mitigate this, most provably efficient learning algorithms rely on the *optimism in the face of uncertainty* principle to incentivize exploration [3]. By artificially inflating the perceived value of every state-action pair with an exploration bonus, an upper-confidence bound (UCB) method encourages the learning agent to visit poorly-understood regions of the state-action space. This bonus term decays as uncertainty is resolved. While model-based UCB approaches like Azar et al. [2] and Zhang et al. [14] have established minimax-optimal regret bounds (up to log factors), their theoretically derived

*Proc. of the Adaptive and Learning Agents Workshop (ALA 2026), Aydeniz, Delgrange, Mohammedalamen, Yang (eds.), May 25 – 26, 2026, Paphos, Cyprus, https://alaworkshop2026.github.io/.* 2026.

bonuses are often overly conservative in practice and remain decoupled from the underlying system dynamics.

Rather than relying on additive bonuses, posterior-sampling-based methods maintain a posterior distribution over possible environmental dynamics. Exploration is therefore naturally guided by the variance of policies sampled from this distribution. These Bayesian-based methods have demonstrated strong empirical performance well before their theoretical properties were fully understood [9, 10]. Notably, this gap remains one decade later. Tiapkin et al. [11]'s empirical evaluations demonstrate that the classic posterior-sampling approach of Osband et al. [9] can outperform the state-of-the-art UCB-based method in Azar et al. [2].

Recent extensions have sought to bridge the theoretical gap by taking a model-free approach: Tiapkin et al. [11] mimic optimism by mixing $Q$-value estimates to balance exploration and exploitation, and Wang et al. [13] extend their work by leveraging ensembles to further calibrate this optimistic bias. Both papers prove a sublinear regret bound of $\tilde{O}\left(\sqrt{H^5SAK}\right)$, where $H$ is the episode length, $S$ is the number of states, $A$ is the number of actions, and $K$ is the number of episodes. In the pursuit of tractable model-free variants, computational compromises have been introduced. While the seminal work in posterior sampling utilized real-time information updates [9], recent literature has introduced "stages" (batched processing of observations) and other techniques that delay the integration of new data. Even nascent frameworks that avoid explicit staging such as Wang et al. [13] still exhibit latency in their update cycles. While these developments represent significant theoretical milestones, we will show in this work that ingesting information *quickly* is key to avoiding algorithmic inefficiencies in practice.

## 3 PROBLEM FORMULATION

We consider a finite, episodic Markov Decision Process (MDP) with horizon $H \in \mathbb{N}$, state space $\mathcal{S}$, and action space $\mathcal{A}$. Episodes are indexed by $k \in [K] := \{1, \ldots, K\}$ and within-episode time by $h \in [H]$. In contrast to standard infinite-horizon formulations where transition dynamics and value functions are time-independent, our finite-horizon setting relies on step-dependent components. Consequently, we denote step-specific variables with a subscript $h$ – such as policies $\pi_h$, rewards $r_h$, transition probabilities $P_h$, and value functions $V_h$ and $Q_h$. At step $h$ of episode $k$, the agent observes $s_{h,k} \in \mathcal{S}$, picks $a_{h,k} \in \mathcal{A}$, receives reward $r_{h,k}(s_{h,k}, a_{h,k}) \in [0, 1]$, and transitions to $s_{h+1,k} \sim P_{h,k}(\cdot \mid s_{h,k}, a_{h,k})$. The episode starts from $s_{1,k}$ and terminates at step $H$. While the reward and transition functions may differ at varying positions within an episode, they are stationary across episodes, i.e., $r_{h,k_1}(\cdot) = r_{h,k_2}(\cdot)$, and similarly for $P_{h,k}(\cdot)$.

A policy is a sequence of decisions $\pi = \{\pi_h\}_{h=1}^H$ where $\pi_h(\cdot \mid s) \in \Delta\mathcal{A}$. For any policy $\pi$, we define the value and action-value functions

$$V_h^\pi(s) = \mathbb{E}\left[\sum_{t=h}^{H} r_t(s_t, a_t) \,\middle|\, s_h = s\right], \tag{1}$$

$$Q_h^\pi(s, a) = r_h(s, a) + \mathbb{E}\left[\sum_{t=h+1}^{H} r_t(s_t, a_t) \,\middle|\, s_h = s, a_h = a\right], \tag{2}$$

respectively, with $V_{H+1}^\pi \equiv 0$. We denote the optimal value functions by $V_h^*$ and $Q_h^*$, and the optimal policy across all episodes by $\pi^* \in \arg\max_\pi V_1^\pi(s_1)$.

The learner's goal is to maximize the cumulative reward over all episodes, or equivalently, minimize the *regret* under a policy $\pi$ over all episodes:

$$\text{Regret}_\pi := \sum_{k=1}^{K} \left(V_1^*(s_1) - V_1^{\pi^k}(s_1)\right). \tag{3}$$

### 3.1 Bayesian-Based Learning Approach

Before proceeding, let us introduce the Bayesian-based model-free learning approach to be used throughout. The learner is *model-free*: it observes $(s_{h,k}, a_{h,k}, r_{h,k}, s_{h+1,k})$ during interaction, but does not estimate $P_{h,k}$ or $r_{h,k}$. Rather, it directly estimates $V_{h,k}^\pi$ so as to minimize regret. To do so, the learner samples an ensemble of $Q$-values according to the posterior distribution of the Bayesian model of the unknown MDP system. `RandomizedQ` is the state-of-the-art algorithm for this problem [13]. Therefore, for clarity, we illustrate our ideas using their algorithm as a basis. To indicate terms associated with *exploratory* estimations, we use $\check{\square}$. To indicate terms associated with *exploitative* estimations, we use $\tilde{\square}$.

## 4 OPPORTUNITIES FOR FASTER LEARNING

In this section, we introduce our core conceptual idea for information utilization: encouraging faster learning rates. We call the combination of these three ideas `ReversedQ`. Building upon `RandomizedQ` [13], `ReversedQ` (detailed in Algorithm 1) integrates three synergistic components: backward passes, accelerated value update schedules, and informed value initialization. Together, these mechanisms allow `ReversedQ` to propagate observed signals more efficiently. This core idea is surprisingly straightforward: in a stationary system such as ours, there is an advantage to utilizing information as soon as it is received.

### 4.1 Backward Pass Through An Episode

By construction, in standard RL, the value function ($V_h$) and action-value function ($Q_h$) at step $h \in [H]$ are conditioned on the expected cumulative reward of subsequent time steps. Updating these functions in a forward chronological order introduces delay: updates at step $h$ rely on values from step $h+1$ that have not yet been updated within the current time step. This results in the propagation of old estimates derived from previous, likely suboptimal, trajectories.

In episodic settings, this lag can be mitigated by leveraging the full observability of rewards upon episode completion. We replace `RandomizedQ`'s forward pass with a backward pass mechanism on line 9. This ensures that value updates utilize the most recent information available from the current trajectory.

### 4.2 Value Update and Restart Timing

We suggest changes to the method of updating action-value function $\check{Q}_i$ and value table $\check{V}_i$, on lines 18 and 19 respectively, and restarting exploration ensemble $\check{Q}_i^j$. These changes allow for agile initial convergence while retaining the adaptability needed to correct for inaccuracies caused by unforeseen failures.

**Algorithm 1:** ReversedQ

**Input:** Inflation coefficient $\kappa$, ensemble size $J$, number of prior transitions $n_0$

**Data:** Visitation counter $N_h(s,a)$; Policy Q-table $Q_h(s,a)$; Q-ensembles $\tilde{Q}_h^j, \breve{Q}_h^j$; value tables $\tilde{V}_h, \breve{V}_h$; mixing rate $\eta \leftarrow \frac{1}{\sqrt{H+1}}$; reward table *rewards*; trajectory table *path*

// Informed value initialization

1 **Initialize:** $V^0 = \{(H-h+1)\}_{h=1}^{H+1}$, $\tilde{V}_h(s), \breve{V}_h(s) \leftarrow V_h^0$; $Q_h(s,a), \breve{Q}_h(s,a), \tilde{Q}_h^j(s,a), \breve{Q}_h^j(s,a) \leftarrow V_{h+1}^0$; $N_h(s,a) \leftarrow 0$; for any $(j,h,s,a) \epsilon [J] \times [H] \times \mathcal{S} \times \mathcal{A}$

2 **for** *episode* $k \leftarrow 1$ **to** $K$ **do**

3 **for** *step* $h \leftarrow 1$ **to** $H$ **do**

4 Take action $a_h \leftarrow \arg\max_a Q_h(s_h, a)$;

5 Observe $s_{h+1}$ and receive reward $r(s_h, a_h)$

6 $rewards_h \leftarrow r_h(s_h, a_h)$, $path_h \leftarrow (s_h, a_h)$;

7 **if** $h = H$ **then**

8 $path[H+1] \leftarrow (s_{h+1}, a_{arbitrary})$

9 **for** $i = H, H-1, \cdots, 1$ **do**

// Traverse backwards on trajectory

10 $s_i, a_i \leftarrow path_i, r \leftarrow rewards_i$;

11 $\tilde{w}^j \sim Beta(\frac{H+1}{\kappa}, \frac{N_i(s_i,a_i)+n_0}{\kappa})$

12 $\breve{w}^j \sim Beta(\frac{1}{\kappa}, \frac{N_i(s_i,a_i)+n_0}{\kappa})$

13 $\tilde{Q}_i^j(s_i,a_i) \leftarrow$ $(1-\tilde{w}^j)\tilde{Q}_i^j(s_i,a_i) + \tilde{w}^j(r + \tilde{V}_{i+1}(s_{i+1}))$

14 $\breve{Q}_i^j(s_i,a_i) \leftarrow$ $(1-\breve{w}^j)\breve{Q}_i^j(s_i,a_i) + \breve{w}^j(r + \breve{V}_{i+1}(s_{i+1}))$

15 $a_{temp} \leftarrow \arg\max_{a \in A} Q_i(s_i, a)$

16 $j_{temp} \leftarrow \arg\max_{j \in [J]} \breve{Q}_i^j(s_i, a_i)$

17 $\tilde{V}_i(s_i) \leftarrow \min\{V^0, \max_{j \in [J]} \tilde{Q}_i^j(s_i, a_{temp})\}$

// Accelerated value updates

18 $\breve{Q}_i(s_i,a_i) \leftarrow \breve{Q}_i^{j_{temp}}(s_i, a_{temp})$

19 $\breve{V}_i(s_i) \leftarrow \max_{a \in A} \breve{Q}_i(s_i, a)$

20 $Q_i(s_i,a_i) \leftarrow \min\{Q_i(s_i,a_i), \Big(\eta \cdot \max_{j \in [J]} \tilde{Q}_i^j(s_i,a_i) + \eta \cdot \breve{Q}_i(s_i,a_i)\Big)\}$

21 $N_i(s_i,a_i) \leftarrow N_h(s_i,a_i) + 1$

22 Reset $rewards[1{:}H], path[1{:}H+1]$

Firstly, staging updates is often beneficial for the purpose of theoretical evaluation of an algorithm. However, it is known that they practically withhold valuable, recent information from the learner [11]. Naturally, we have allowed the agent to access this information as it becomes available by letting $\breve{Q}_i(s_i,a_i)$ and $\breve{V}_i(s_i)$ update per time step (lines 18 and 19).

Secondly, in order to induce and maintain optimism, `RandomizedQ` periodically resets the optimistically-driven exploration ensemble, $\breve{Q}_i^j$. While periodic resets can improve adaptability to account for the probability of failure, we have found that indiscriminate resetting hinders learning by discarding valid information, forcing the policy to re-converge at the cost of sample efficiency. In order to balance optimism, we have chosen to no longer reset the exploration ensemble, but to increase the rate at which it is optimistically mixed – from $(1-\eta)$ to $\eta$ on line 20. This allows the agent to continue to benefit from increased optimism without suffering periodic losses.

Finally, we note that the upper-bound clipping via $\min(\cdot)$ on lines 17 and 20 are inherited from Wang et al. [12, 13]'s implementation, where they prevent value estimates from exceeding the initialized upper bound. Although absent in `RandomizedQ`'s pseudocode, they are present in the released code and carried forward unchanged.

## 4.3 Value Initialization

A common technique in online RL is the optimistic initialization of $Q$ and $V$ values, which are subsequently refined based on observations. Within the `RandomizedQ` framework, value resets occur periodically; specifically, the exploratory $\breve{Q}$-ensembles are re-initialized "to mitigate outdated data and ensure optimism" [13]. Unfortunately, Wang et al. [13] employ an initial value that exceeds the bounds of the environment to ensure their theoretical bound on regret (Theorem 1). Introducing unnecessary slack reduces learning efficiency. Therefore, we suggest an initial value estimation of $H-h+1$ instead of $2(H-h+1)$ (line 1). As we will demonstrate in the following section, even this minor change has a big impact.

# 5 EMPIRICAL EVALUATION

To evaluate the practical impact of our framework, we empirically evaluate `ReversedQ` on two MDP systems. All experiments were conducted with the following hyperparameters: number of ensembles $J = 10$, number of prior transitions $n_0 = \frac{1}{S}$, and inflation coefficient $\kappa = 1$. The tests were performed on a machine with an Intel i7-13700K (3.4 GHz) processor and 64 GB of RAM.

We analyze `ReversedQ` and the three components individually, each termed `RandomizedQ-Backward`, `RandomizedQ-Update`, and `RandomizedQ-Init`, against `RandomizedQ`.

In BDCL, the agent can take $\mathcal{A} = 5$ actions with fail probability $p_{fail} = 0.02$ over an episode of length $H = 5$ and total horizon $T = 2,500$. We compute 20 simulation seeds for each policy evaluation. In the chain MDP, we compute 5 simulation seeds of episode length $H = 50$, horizon $T = 60,000$, and $\mathcal{S} = 20$ states.

*Metric.* To measure performance, we compare the scaled mean cumulative rewards, where scaling uses the mean cumulative reward of an Oracle policy as the upper bound and a Random policy as the lower bound:

$$\text{Scaled}(\pi) = 100 \cdot \frac{\mathbb{E}[R(\pi)] - \mathbb{E}[R(Random)]}{\mathbb{E}[R(Oracle)] - \mathbb{E}[R(Random)]} \tag{4}$$

We report 95% confidence intervals over independent runs.

## 5.1 Environments

*Bidirectional Diabolical Combination Lock* (BDCL) [1] is an episodic MDP designed to be hard to explore. The MDP has four states: **Start**, **Sink** (absorbing), **Lock 1**, and **Lock 2**. Each episode begins in **Start**. Let the action set be $\{0, \ldots, A-1\}$ and define $\tau = \lfloor (A-1)/2 \rfloor$; actions $0, \ldots, \tau$ route the first step to **Lock 1**, while actions $\tau+1, \ldots, A-1$ route to **Lock 2** (this handles both even and odd A). Once on **Lock** $i$ at any $t \in \{1, \ldots, H-1\}$, exactly one randomly selected "progress"

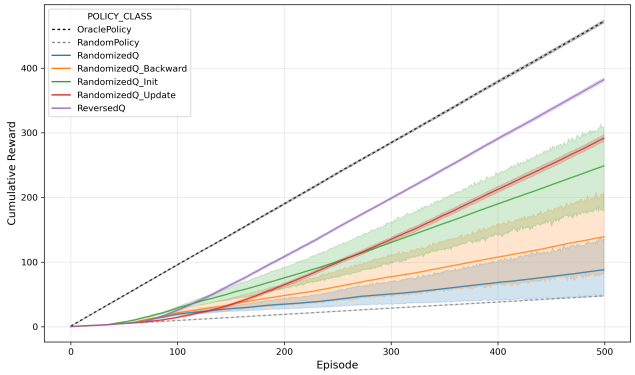


**Figure 1: Agents face difficulty exploring BDCL as they encounter several significant, yet suboptimal rewards before exploration to the terminal state in the higher-yielding lock. This creates wide confidence intervals in cumulative rewards across several simulations. However, there is little variance in the reward that `ReversedQ` achieves, indicating robustness: the policy consistently finds the optimal solution, regardless of slight simulation-to-simulation deviations. Furthermore, an analysis of our individual contributions demonstrates that each component independently benefits algorithmic performance, consistently yielding higher rewards than the baseline, `RandomizedQ`.**

**Table 1: BDCL Scaled Mean Cumulative Reward with 95% Confidence Intervals**

| *Policy* | *Scaled Mean Reward* (%) |
|---|---|
| RandomizedQ | 9.53 ± 11.04 |
| RandomizedQ-Backward | 21.55 ± 15.97 |
| RandomizedQ-Init | 47.36 ± 16.6 |
| RandomizedQ-Update | 57.44 ± 1.32 |
| **ReversedQ** | **78.78 ± 0.75** |

action advances along that lock; any other action transitions to **Sink**, which is absorbing. Rewards are zero in **Start** and on intermediate lock steps. At $t = H$, a terminal reward $R_i$ is obtained if the agent remains on **Lock** $i$. The second lock yields higher reward than the first lock ($r_{l1} = 0.25$, $r_{l2} = 1.0$). The **Sink** yields $r_{sink} = \frac{1}{8H}$ per step, so an agent spending the entire episode there receives at most $\frac{1}{8} < r_{l1} < r_{l2}$. A failure mechanism with probability $p_{fail} \in [0, 1)$ overrides the chosen action and sends the agent to **Sink** regardless of correctness. The identity of each lock's unique progress action is unfixed and changes across time. Collectively, these design choices make the environment exploration-hard: rewards are sparse and delayed, the progress action is unique at each step, and any deviation – or failure with probability $p_{fail}$ – irreversibly transitions the agent to the absorbing **Sink**.

We also compare against the *chain MDP* environment used in the empirical results of Tiapkin et al. [11], Wang et al. [13]. The agent begins in the leftmost state $s_1$. At each step, the agent chooses

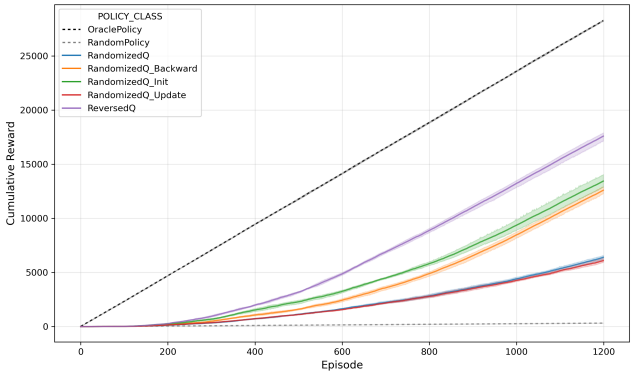


**Figure 2: `ReversedQ` converges in fewer episodes and achieves higher reward than `RandomizedQ`. Backward passes and informed initialization each improve performance, while the value update and restart timing changes perform similarly to baseline.**

**Table 2: Chain Scaled Mean Cumulative Reward with 95% Confidence Intervals**

| *Policy* | *Scaled Mean Reward* (%) |
|---|---|
| RandomizedQ | 21.76 ± 1.11 |
| RandomizedQ-Backward | 43.93 ± 1.94 |
| RandomizedQ-Init | 46.95 ± 3.41 |
| RandomizedQ-Update | 20.67 ± 1.13 |
| **ReversedQ** | **61.81 ± 2.23** |

an action $a \in \{\textbf{left}, \textbf{right}\}$. The chosen action succeeds with probability $p = 0.9$, moving the agent in the intended direction, and fails with probability $1 - p$, moving the agent in the opposite direction. Shifting into a boundary maintains the agent's current position with probability $p$ and moves the agent inward with probability $1 - p$. Rewards are sparse: the agent receives rewards only at the boundary states. The starting state $s_1$ yields suboptimal reward $r = 0.05$ and the terminal state $s_S$ grants a large payoff of $r = 1$.

## 5.2 Results and Discussion

Our empirical analysis has shown that mitigating theoretically conservative mechanisms enables `ReversedQ` to achieve significantly higher rewards compared to `RandomizedQ` across both environments. As seen in Figures 1 and 2, `ReversedQ` takes significantly fewer episodes, and therefore samples, to converge on the optimal policy when compared against `RandomizedQ` and each isolated component implemented on `RandomizedQ`.

*Backward pass through an episode.* Individual and combined evaluation of this property has shown a significant increase in reward in each scenario. When this change is implemented independently with our baseline algorithm, `RandomizedQ`, the two together control over-exploration and reduce regret. This leads to hastened convergence and demonstrates that stationary performance can be

substantially improved when optimism is calibrated and information from previous steps is fully utilized. In BDCL (Figure 1) and chain (Figure 2) environments, the backward pass property alone improves the scaled reward of RandomizedQ from 9.53% ± 11.04 to 21.55% ± 15.97 and 21.76% ± 1.11 to 43.93% ± 1.94, respectively.

*Update and restart timing.* Optimized update and restart frameworks allow for better rate and retention of learning. Figure 1 supports this hypothesis, showing improvement in scaled reward from 9.53% ± 11.04 to 57.44% ± 1.32 in BDCL. However, this change does not significantly impact RandomizedQ individually in the chain environment, yielding roughly baseline performance. The asymmetry between environments is explained by their structure. In BDCL, stale value estimates are particularly costly because a single suboptimal action irreversibly transitions the agent to the absorbing sink state. Prompt updates therefore have a sizable impact. In the chain environment, the reward structure is less complex. Consequently, delayed updates are less detrimental, explaining why this modification alone does not significantly improve over baseline. Despite its limited independent impact in the chain environment, the optimized timing framework is critical; it allows for prompt reactions to recent discoveries and the retention of learning across episodes, which ultimately fosters faster, more stable convergence as seen across the figures.

*Value Initialization.* Informed value initialization produces a large and consistent performance gain across both environments. In BDCL, replacing RandomizedQ's overly optimistic initialization with the tighter bound $V_h^0 = H - h + 1$ improves scaled mean reward from 9.53% ± 11.04 to 47.36% ± 16.6 (Table 1), indicating that a substantial fraction of simulations accumulate higher rewards much earlier than the baseline. In the chain MDP, the same modification increases performance from 21.76% ± 1.11 to 46.95% ± 3.41 (Table 2). These improvements suggest that RandomizedQ's original initialization is overly optimistic relative to reachable returns. Correcting excessive optimism can take many episodes, delaying reliable exploration. Initializing at a realistic upper bound keeps beneficial optimism while reducing unnecessary slack, leading to faster calibration.

## 6 CONCLUSION

In this work, we show that accelerating the feedback loop between observation and estimation significantly improves posterior-sampling-based online RL. By incorporating backward passes, accelerated value update schedules, and informed value initialization into RandomizedQ, we achieve substantial performance gains across both environments. Specifically, our combined approach, ReversedQ, increased the scaled mean cumulative reward from 9.53% to 78.78% in the BDCL environment and from 21.76% to 61.81% in the chain MDP environment. Furthermore, our ablation studies confirm that each isolated mechanism independently accelerates convergence and reduces regret.

The empirical performance of ReversedQ substantially exceeds that of RandomizedQ [13] across both tested environments. The established regret bound of $\tilde{O}(\sqrt{H^5SAK})$ for RandomizedQ [13] represents a worst-case theoretical ceiling; the gap between such bounds and empirical performance aligns with existing observations that posterior-sampling algorithms frequently outperform their theoretical guarantees [10]. Rather than merely reporting this gap, we propose specific algorithmic modifications grounded in a structural explanation of their performance. Moving forward, we would like to formally prove that RandomizedQ's regret bounds are preserved under these modifications. Furthermore, this framework facilitates the isolation of stationary online RL dynamics from external factors such as non-stationarity or multi-agent interactions. Recent work by Nonaka et al. [8] supports this view: the authors discuss modifying a stationary algorithm for use in a non-stationary setting (indeed, they use RandomizedQ for illustration).

## ACKNOWLEDGMENTS

We gratefully acknowledge Hiroshi Nonaka, Simon Ambrozak, and Amedeo Ercole for their invaluable insights and support. We also thank Dr. William Gasarch and the REU-CAAR cohort at the University of Maryland for fostering the collaborative research environment and providing the facilities that made this work possible. Finally, we extend our gratitude to Red Cell Partners for their sponsorship of Aviva Prins.

## REFERENCES

[1] Alekh Agarwal, Mikael Henaff, Sham Kakade, and Wen Sun. 2020. Pc-pg: Policy cover directed exploration for provable policy gradient learning. *Advances in neural information processing systems* 33 (2020), 13399–13412.

[2] Mohammad Gheshlaghi Azar, Ian Osband, and Rémi Munos. 2017. Minimax regret bounds for reinforcement learning. In *International conference on machine learning*. PMLR, 263–272.

[3] Thomas Jaksch, Ronald Ortner, and Peter Auer. 2010. Near-optimal Regret Bounds for Reinforcement Learning. *Journal of Machine Learning Research* 11 (2010), 1563–1600.

[4] Chi Jin, Zeyuan Allen-Zhu, Sebastien Bubeck, and Michael I Jordan. 2018. Is Q-learning provably efficient? *Advances in neural information processing systems* 31 (2018).

[5] J. Kober, J. Andrew (Drew) Bagnell, and J. Peters. 2013. Reinforcement Learning in Robotics: A Survey. *International Journal of Robotics Research* 32, 11 (September 2013), 1238 – 1274.

[6] Anubha Mahajan, Shreya Hegde, Ethan Shay, Daniel Wu, and Aviva Prins. 2025. Comparative Analysis of Multi-Agent Reinforcement Learning Policies for Crop Planning Decision Support. *AAAI Workshop on Cooperative Multi-Agent Systems Decision-making and Learning (CMASDL)* (2025).

[7] Weichao Mao, Kaiqing Zhang, Ruihao Zhu, David Simchi-Levi, and Tamer Başar. 2024. Model-Free Nonstationary Reinforcement Learning: Near-Optimal Regret and Applications in Multiagent Reinforcement Learning and Inventory Control. *Management Science* (2024).

[8] Hiroshi Nonaka, Simon Ambrozak, Sofia R. Miskala-Dinc, Amedeo Ercole, and Aviva Prins. 2025. Efficient Restarts in Non-Stationary Model-Free Reinforcement Learning. In *NeurIPS 2025 Workshop: Second Workshop on Aligning Reinforcement Learning Experimentalists and Theorists*.

[9] Ian Osband, Daniel Russo, and Benjamin Van Roy. 2013. (More) efficient reinforcement learning via posterior sampling. *Advances in Neural Information Processing Systems* 26 (2013).

[10] Ian Osband and Benjamin Van Roy. 2017. Why is posterior sampling better than optimism for reinforcement learning?. In *International conference on machine learning*. PMLR, 2701–2710.

[11] Daniil Tiapkin, Denis Belomestny, Daniele Calandriello, Eric Moulines, Remi Munos, Alexey Naumov, Pierre Perrault, Michal Valko, and Pierre Ménard. 2023. Model-free posterior sampling via learning rate randomization. *Advances in Neural Information Processing Systems* 36 (2023), 73719–73774.

[12] He Wang, Xingyu Xu, and Yuejie Chi. 2025. RandomizedQ. Retrieved April 15, 2026 from https://github.com/IrisHeWANG/RandomizedQ Commit: 763723c.

[13] He Wang, Xingyu Xu, and Yuejie Chi. 2026. Provably Efficient and Agile Randomized Q-Learning. In *The 29th International Conference on Artificial Intelligence and Statistics*. https://arxiv.org/abs/2506.24005

[14] Zihan Zhang, Yuxin Chen, Jason Lee, and Simon S Du. 2025. Settling the Sample Complexity of Online Reinforcement Learning. *J. ACM* 72, 3 (2025), 1–63.