Efficient Restarts in Non-Stationary Model-Free Reinforcement Learning

Hiroshi Nonaka *
Soka University of America
hnonaka@soka.edu

Simon Ambrozak *
University of Maryland, College Park
sambroza@umd.edu

Sofia R. Miskala-Dinc †
University of Maryland, College Park
smiskala@umd.edu

Amedeo Ercole †
University of Maryland, College Park
aercole@umd.edu

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

Abstract

In this work, we propose three efficient restart paradigms for model-free non-stationary reinforcement learning (RL). We identify two core issues with the restart design of Mao et al. (2022)'s RESTARTQ-UCB algorithm: (1) complete forgetting, where all the information learned about an environment is lost after a restart, and (2) scheduled restarts, in which restarts occur only at predefined timings, regardless of the incompatibility of the policy with the current environment dynamics. We introduce three approaches, which we call *partial*, *adaptive*, and *selective* restarts to modify the algorithms RESTARTQ-UCB and RANDOMIZEDQ (Wang et al., 2025). We find near-optimal empirical performance in multiple different environments, decreasing dynamic regret by up to 91% relative to RESTARTQ-UCB.

1 Introduction

Reinforcement learning (RL) is a computational approach to solving interactive problems, such as crop management, inventory control, and board games (Gautron et al., 2022; Lu and Prins, 2023; Mahajan et al., 2025; Tao et al., 2023; Mao et al., 2022; Silver et al., 2016, 2017). While conventional RL assumes stationarity, many real-life settings present more complex dynamics called non-stationarity, in which reward functions and transitions change over time. To model these real-world scenarios, we focus on reinforcement learning in episodic non-stationary Markov decision process (MDP) structures. These environments pose unique challenges for learning.

RL for non-stationary environments is difficult because algorithms must manage two key trade-offs: *exploration versus exploitation* and *remembering versus forgetting*. The exploration-exploitation trade-off is inherent to RL. This trade-off becomes even more challenging in non-stationary environments. Since the environment is changing, an RL algorithm must explore more in order to account for change, but it loses out on reward by not exploiting during that time. In regards to remembering versus forgetting, an agent must decide what data to keep and what to discard, since previously acquired information about the environment may no longer be valid.

*These authors contributed equally to this work
†These authors contributed equally to this work

2nd Workshop on Aligning Reinforcement Learning Experimentalists and Theorists (ARLET 2025).

Mao et al. (2022) introduce a model-free algorithm called RESTARTQ-UCB, which manages these trade-offs in such a way that promises near-optimal performance. It uses an optimism term to encourage exploration and performs an occasional *restart* that forgets all previously seen observations. Despite an impressive asymptotic performance guarantee, we find that fully erasing all learned data at scheduled intervals does not best utilize the information that the agent has observed.

To address these inefficiencies, we discuss three paradigms, which we call *partial*, *adaptive*, and *selective* restarts. Partial restarts address the issue of complete forgetting by resetting the Q-table to a tighter upper bound than the loose bound that is commonly used. Adaptive restarts identify when restarts are most needed, rather than relying on a fixed hyperparameter. Finally, selective restarts combine these two approaches and reset only a subset of Q-table entries to further accelerate convergence to a new optimum.

We evaluate our approaches by comparing cumulative reward in two environments: a new randomized environment that we call RandomMDP and Bidirectional Diabolical Combination Locks (BDCL) (Agarwal et al., 2020). Our empirical results are impressive. We observe a 74% and 91% decrease in dynamic regret in RandomMDP and BDCL environments, respectively. The latter result is particularly impressive because the BDCL environment is designed to be difficult to explore. These results demonstrate the potential of our frameworks to overcome the restart inefficiencies. We hope that by adding our restart modifications to a theoretically robust algorithm, we can achieve near-optimal performance in practice while maintaining the spirit of RESTARTQ-UCB's asymptotic guarantees.

2 Related work

Conventional RL studies consider stationary environments. Those foundational works include value-based methods, such as Q-LEARNING (Watkins and Dayan, 1992) and DEEP Q-LEARNING (Mnih et al., 2013), and policy-gradient and actor-critic methods, which include REINFORCE (Williams, 1992) and ACTOR-CRITIC (Konda and Tsitsiklis, 1999). These algorithms demonstrate good performance in various stationary MDPs. Nonetheless, many real-world scenarios involve non-stationary dynamics, which poses a fundamental limitation to the RL algorithms (da Silva et al., 2006; Gautron et al., 2022; Zhou et al., 2024; Mao et al., 2022). The central challenge, therefore, is to enable RL agents to continuously adapt to these unpredictable and time-varying conditions. Many works introduce model-based RL approaches to solve the non-stationarity issues. Gajane et al. (2019) propose Variation-aware UCRL, a variant of the UCRL algorithm that restarts according to a schedule dependent on the total variation in the MDP. Cheung et al. (2020) introduce the Sliding Window UCRL2 with Confidence Widening (SWUCRL2-CW) algorithm, which uniquely addresses challenges posed by conventional optimistic exploration techniques in non-stationary MDPs by incorporating additional optimism through a confidence widening technique. da Silva et al. (2006) approach this problem by introducing a multiple-model approach called RL-CD, where it evaluates the prediction quality of several partial models and incrementally builds new ones as needed, selecting the most appropriate one when a context change is detected.

While most approaches have been model-based, they typically suffer from time and space complexity (Jin et al., 2018; Mao et al., 2022; Zhang et al., 2020). This has motivated a growing interest in model-free approaches, which offer advantages in terms of online applicability and flexibility. Mao et al. (2022) developed RESTARTQ-UCB, which is a pioneering model-free non-stationary RL for episodic MDPs, achieving competitive dynamic regret by adopting a simple restarting strategy and incorporating an extra optimism term.

3 Preliminaries

We consider a non-stationary finite-horizon episodic Markovian setting. An instance of a Markov decision process (MDP) is given by $\mathcal{M} = (\mathcal{S}, \mathcal{A}, H, P, r)$, where $\mathcal{S}$ is a finite set of states, $\mathcal{A}$ is a finite set of actions, $P = \{P_h^m\}_{h\in[H],m\in[M]}$ is a set of transition kernels, and $r = \{r_h^m\}_{h\in[H],m\in[M]}$ is a set of reward functions. The setting contains M episodes, each of length H. We define T as the total number of steps in the entire horizon, where $T = MH$ and $M, H, T \in \mathbb{N}$. For $N \in \mathbb{N}$, we use the notation convention $[N] \stackrel{\text{def}}{=} [1, 2, \ldots, N]$. We denote $S = |\mathcal{S}|$ and $A = |\mathcal{A}|$. An agent observes a state $s_h^m \in \mathcal{S}$ and takes an action $a_h^m \in \mathcal{A}$ by following a policy function. The

environment returns a reward $r_h^m(s_h^m, a_h^m) \in [0, 1]$. The environment then transitions to a new state $s_{h+1}^m \sim P_h^m(\cdot \mid s_h^m, a_h^m)$. Since the environment is non-stationary, P_h^m and r_h^m may vary over time with respect to m and h.

The degree of non-stationarity of a given MDP is quantified via *variation budget*. This is defined as the sum of the supremum distances between rewards or transition probabilities across two consecutive episodes:

$$\Delta_r \stackrel{\text{def}}{=} \sum_{m=1}^{M-1} \sum_{h=1}^{H} \sup_{s,a} \left| r_h^m(s, a) - r_h^{m+1}(s, a) \right| \tag{1}$$

$$\Delta_p \stackrel{\text{def}}{=} \sum_{m=1}^{M-1} \sum_{h=1}^{H} \sup_{s,a} ||P_h^m(\cdot \mid s, a) - P_h^{m+1}(\cdot \mid s, a)||_1, \tag{2}$$

where $|| \cdot ||_1$ denotes L^1-norm. Δ_r and Δ_p represent the total amount of change in reward functions and transitions episode to episode over the entire horizon, thus indicating the extent of non-stationarity in the environment.

The goal of the RL agent is to solve for a policy of actions π, defined by the collection of functions $\pi_h^m : \mathcal{S} \to \mathcal{A}$. Thus, $a_h^m = \pi_h^m(s_h^m)$. Under this finite setting, there exists an optimal policy $\pi^\star$ that maximizes total reward. A state value function $V_h^{m,\pi} : \mathcal{S} \to \mathbb{R}$ returns a scalar quantifying the value of a state at episode m and step h under a policy π:

$$V_h^{m,\pi} \stackrel{\text{def}}{=} \mathbb{E}\left[\sum_{h'=h}^{H} r_{h'}^m(s_{h'}, \pi_{h'}^m(s_{h'})) \mid s_h = s, s_{h'+1} \sim P_h^m(\cdot \mid s_{h'}, a_{h'}) \right]$$

Likewise, an action value function $Q_h^m : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ returns the value of a state-action pair under a policy:

$$Q_h^m(s, a) \stackrel{\text{def}}{=} r_h^m(s, a) + \mathbb{E}\left[\sum_{h'=h+1}^{H} r_{h'}^m(s_{h'}, \pi_{h'}^m(s_{h'})) \mid s_h = s, a_h = a, s_{h'+1} \sim P_h^m(\cdot \mid s_{h'}, a_{h'}) \right]$$

Analogously, the state and action value functions of $\pi^\star$ are given by $V_h^{m,\star}(s) = \sup_\pi V_h^m(s)$ and $Q_h^{m,\star}(s, a) = \sup_\pi Q_h^m(s, a)$. Note that in this episodic setting we may let $V_{H+1}^{m,\pi}(s) = 0, \forall s \in \mathcal{S}, m \in [M]$.

The goal of the agent is to maximize total cumulative reward. *Dynamic regret* is the cumulative difference between the optimal and policy-based state values per episode:

$$\mathcal{R}(\pi, M) \stackrel{\text{def}}{=} \sum_{m=1}^{M} \left[V_1^{m,\star}(s_1^m) - V_1^{m,\pi}(s_1^m) \right].$$

Since dynamic regret measures the amount of reward that the agent missed out on each episode, relative to a (fixed, but possibly unknown) optimal policy, maximizing total cumulative reward is analogous to minimizing dynamic regret.

3.1 Limitations of existing approaches

RESTARTQ-UCB is a Q-learning algorithm that utilizes upper confidence bounds and scheduled restarts that reset the learned Q-values every time the end of an *epoch* is reached. An epoch is a sequential group of episodes, and any learning in one epoch does not affect learning in another because RESTARTQ-UCB is completely restarted when an epoch ends. The number of epochs is set to $D = S^{-\frac{1}{3}} A^{-\frac{1}{3}} \Delta^{\frac{2}{3}} H^{-\frac{2}{3}} T^{\frac{1}{3}}$, and the number of episodes in each epoch is $K = \lceil \frac{M}{D} \rceil$. The algorithm is reproduced in Algorithm 1. Although RESTARTQ-UCB has a near-optimal upper bound on dynamic regret $\tilde{O}\left(S^{\frac{1}{3}} A^{\frac{1}{3}} (\Delta_r + \Delta_p)^{\frac{1}{3}} H T^{\frac{2}{3}} \right)$, our empirical analysis of its performance reveals a significant gap between theory and practice. Most MDPs will have some exploitable structure that we seek to take advantage of while not losing performance guarantees in the worst-case scenarios.

Algorithm 1: RestartQ-UCB (Hoeffding), Mao et al. (2022)

1 **for** *epoch* $d \leftarrow 1$ *to* D **do**

2 **Initialize:** $V_h(s) \leftarrow H-h+1$, $Q_h(s,a) \leftarrow H-h+1$, $N_h(s,a) \leftarrow 0$, $\check{N}_h(s,a) \leftarrow 0$, $\check{r}_h(s,a) \leftarrow 0$, $\check{v}_h(s,a) \leftarrow 0$, $\check{\mu}_h(s,a) \leftarrow 0$, $\check{\sigma}_h(s,a) \leftarrow 0$, $\mu_h^{\text{ref}}(s,a) \leftarrow 0$, $\sigma_h^{\text{ref}}(s,a) \leftarrow 0$, $V_h^{\text{ref}}(s) \leftarrow H$, for all $(s,a,h) \in \mathcal{S} \times \mathcal{A} \times [H]$.

3 **for** *episode* $k \leftarrow (d-1)K+1$ *to* $\min\{dK, M\}$ **do**

4 **observe** s_1;

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

6 **Take** action $a_h \leftarrow \arg\max_a Q_h(s_h,a)$; **Receive** reward $R_h(s_h,a_h)$ and observe s_{h+1} $\check{r}_h(s_h,a_h) \leftarrow \check{r}_h(s_h,a_h) + R_h(s_h,a_h)$, $\check{v}_h(s_h,a_h) \leftarrow \check{v}_h(s_h,a_h) + V_{h+1}(s_{h+1})$. $N_h(s_h,a_h) \leftarrow N_h(s_h,a_h)+1$, $\check{N}_h(s_h,a_h) \leftarrow \check{N}_h(s_h,a_h)+1$

7 **if** $N_h(s_h,a_h) \in \mathcal{L}$ **then**

8 $b_h \leftarrow \sqrt{\frac{H^2}{\check{N}_h(s_h,a_h)}\,\iota} + \sqrt{\frac{1}{\check{N}_h(s_h,a_h)}\,\iota}$, $b_\Delta \leftarrow \Delta_r^{(d)} + H \cdot \Delta_p^{(d)}$

9 $Q_h(s_h,a_h) \leftarrow \min\left\{\frac{\check{r}_h(s_h,a_h)}{\check{N}_h(s_h,a_h)} + \frac{\check{v}_h(s_h,a_h)}{\check{N}_h(s_h,a_h)} + b_h + 2b_\Delta, Q_h(s_h,a_h)\right\}$

10 $V_h(s_h) \leftarrow \max_a Q_h(s_h,a)$

11 $\check{N}_h(s_h,a_h) \leftarrow 0$, $\check{r}_h(s_h,a_h) \leftarrow 0$, $\check{v}_h(s_h,a_h) \leftarrow 0$

We identify two sources of learning inefficiency in RESTARTQ-UCB that stem from its restart design. (1) **Complete forgetting**: RESTARTQ-UCB initializes the Q-table to the theoretical maximum value at every restart after an epoch, which requires learning from scratch every time and inflates dynamic regret. (2) **Scheduled restarts**: RESTARTQ-UCB restarts only at the predetermined timings regardless of whether the learned policy needs a restart, that is, whether the policy is incompatible with the current dynamics of the environment or not. To deal with these practical issues, we introduce three restart algorithmic frameworks in the following section, offering more granular control and potentially superior performance by minimizing unnecessary exploration and computational overhead.

4 Proposed approach: towards more efficient convergence

In this section, we introduce partial, adaptive, and selective restarts, which we develop to address the issues of complete forgetting and scheduled restarts.

4.1 Partial restarts

When RESTARTQ-UCB performs a restart at the end of an epoch, it fully resets all learned values, and all the information about the environment is forgotten. However, if the agent has knowledge of the environment's variation budgets Δ_p and Δ_r (Equations 1 and 2), it is possible to solve for a tighter upper bound. We call this a *partial* restart, because, at a reset, each Q-value is raised to some value that is lower than their theoretical maximums.

The goal of a restart in RESTARTQ-UCB is for each Q-value in the algorithm's Q-table to be larger than its optimal Q-value, $Q^\star$. RESTARTQ-UCB achieves this by setting each Q-value to the theoretical maximum, $H-h+1$ (line 2 of Algorithm 1). In this way, the Q-values are sure to be greater than $Q^\star$, but generally are much greater than necessary and cause inefficient convergence. Partial restarts use two pieces of information: the learned Q-value at the end of the epoch, and the maximum theoretical difference in $Q^\star$-values given Δ_r and Δ_p. By adding the difference to the learned Q-value, we can partially restart it in an efficient way.

Δ_p and Δ_r can be used to describe a maximum possible difference in $Q^\star$-values at two different episodes, represented by $Q_h^{k_2,\star}(s,a) - Q_h^{k_1,\star}(s,a)$. Building off of Lemma 1 from Mao et al. (2022), we can show the following:

Lemma 1. *For any triple* (s,a,h) *and any episodes* $k_1, k_2 \in [K]$, *it holds that* $\left|Q_h^{k_2,\star}(s,a) - Q_h^{k_1,\star}(s,a)\right| \le \Delta_r + \frac{1}{2}\Delta_p \min_{k \in k_1,k_2}\left[\max_{s,a,h'>h}\left[Q_h^{k,\star}(s,a)\right]\right]$.

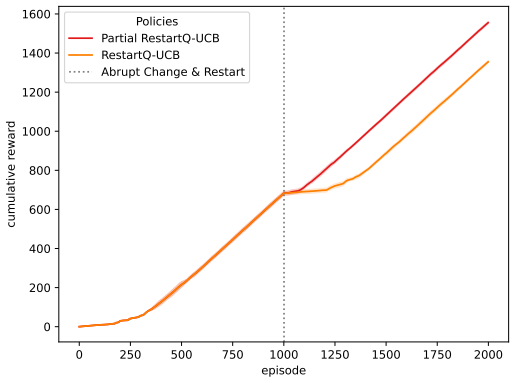

Figure 1: This figure compares the impact of a partial restart (red) as opposed to a full restart (orange), when both are positioned to align with an abrupt change in BDCL. After the abrupt change and restarts at episode 1001, partial restarts allow for much faster learning than full restarts.

The proof is similar to the method used by Mao et al. (2022), using backwards induction on h, the most significant difference is bounding the difference in transition probabilities algebraically instead of using Hölder's inequality. The full proof is provided in Appendix A.

By adding the above bound on the $Q^\star$-value difference to every learned Q-value, we can achieve a successful restart while retaining some environmental information given two assumptions:

1. Q-UCB learning requires that for all (s, a, h) triples, $Q_h(s, a) \geq Q_h^\star(s, a)$ immediately after the restart, where $Q_h(s, a)$ is the learned Q-value for (s, a, h).
2. A learned Q-value for a (s, a, h) triple will never go below the lowest optimal Q-value for that triple during an epoch. Equivalently: $Q_h^K(s, a) \geq \min_{k\in[K]} Q_h^{k,\star}(s, a)$, where $Q_h^K(s, a)$ is the learned Q-value at the end of the epoch, before the restart occurs.

We expect this modification to have the greatest effect in environments with very sparse rewards, such as BDCL, because learned Q-values may become very low compared to their theoretical maximums. In such cases, partial restarts allow for much faster learning and convergence to the optimal policy than a full restart. We give an example of this in Figure 1. In this example, the partial variant recovers faster from an abrupt change in the dynamics of the system. Since the calculations only need to run once per restart, it has a negligible effect on time and space complexity of RESTARTQ-UCB.

4.2 Adaptive restarts

RESTARTQ-UCB restarts at the beginning of a new epoch, which is calculated using S, A, Δ, H, and T. While these timings are chosen to ensure the dynamic regret bound, in practice, scheduled restarts occur at times when a restart is unnecessary. Rather, the agent ought to be using its current policy to maximize reward. To address this problem, we introduce *adaptive* restarts, which detect change in the environment by looking at cumulative reward.

Adaptive restarts estimate how much reward will be gained if a restart happens immediately, and how much reward will be gained if no restart happens. This is done using a sliding window approach over the total reward gained per episode. We note that our use of the term "sliding window" is different from Cheung et al. (2020). The main idea is that if all of the rewards in the window are summed up, we can keep track of the lowest total reward, highest total reward, and current total reward gained in that window. The lowest total reward will be the reward gained during learning (if our algorithm is running as intended). The highest total reward will be the reward we can gain if we know the environment well, and the current total reward is the reward we are currently gaining.

First, we need to decide the length of the sliding window. Since we want to keep track of how much reward we gain during learning, the window length W should be the number of episodes it took for the agent to begin exploiting the best path it has found. W is computed based on Q-table updates while RESTARTQ-UCB begins running.

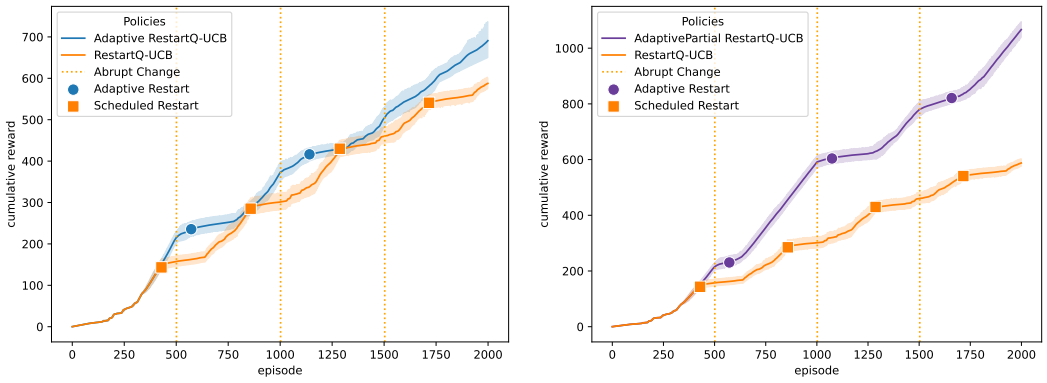

Figure 2: This figure demonstrates that adaptive restarts (blue) perform better than scheduled restarts (orange) in BDCL. On the left are adaptive restarts and scheduled restarts, showing that adaptive restarts only occur after each abrupt change and achieve a higher cumulative reward. This effect is further shown on the right, where RESTARTQ-UCB with adaptive and partial restarts (purple) receives nearly twice as much total reward as scheduled, full restarts.

To find W, we keep track of how often the optimal action changes when the agent attempts to update its Q-table on line 9 of Algorithm 1. If an update is triggered (that is, $N_h(s,a) \in \mathcal{L}$) and the optimal action at that state-timestep does not change (i.e., $\arg\max_a Q_h(s,a)$ stays the same), we count it as one "non-update". Similarly, if the optimal action *does* change, we count it as one "true-update". If H^2 "true-updates" happen before H^2 "non-updates" occur, then the agent is still learning, and both counters are reset to 0. If H^2 "non-updates" happen before H^2 "true-updates", then learning is considered done, and the window length W is set to the difference between the current episode index and the episode index when the update counters were last reset.

Now, we sum over the first W episodes to get our total reward gained during learning, r_L. Every time we reach the end of an episode, we sum over the last W episodes to get our current reward r_C, and the highest value of r_C seen is the best reward gained, r_B.

Since we are in a finite-horizon setting, we assume the agent has knowledge of the simulation horizon T. Using this, we estimate the total reward we will receive if we *do not* restart: $r_C \frac{T-t}{HW}$, where t is the current timestep, and the reward we will receive if we *do* restart: $r_L + r_B(\frac{T-t}{HW} - 1)$.

Therefore, if the following equality is true,

$$r_C \frac{T-t}{HW} < r_L + r_B(\frac{T-t}{HW} - 1)$$

then a restart occurs. Full pseudocode can be found in Appendix B.

In Figure 2, we give an example to demonstrate that adaptive and partial restarts can be used in combination to efficiently overcome the challenges posed a rapidly changing environment. Empirically, RESTARTQ-UCB with adaptive restarts (blue) outperforms scheduled restarts (orange), on the left. This is seen with an even greater effect in the graph on the right, which shows RESTARTQ-UCB with partial and adaptive restarts (purple).

Because we are only keeping track of three values: r_C, r_L, rand r_B, and only calculating a running sum of rewards, the time complexity impact is negligible, and the space needed is at worst $O(T)$.

4.3 Selective restarts

Finally, we introduce selective restarts, wherein we selectively restart only certain (s,a) positions in the Q-table using the upper bound between optimal action values from Lemma 1. The timing of restarts is also adaptively determined. In this respect, selective restarts are a combined approach of partial and adaptive restarts, but tailored to updating only some entries.

In our implementation, a selective restart updates $Q_{h'}(s_{h'}, a_{h'})$ associated with the experiential trajectory to (s, a, h): $\mathcal{T}_h(s, a) \stackrel{\text{def}}{=} \{(h', s_{h'}, a_{h'}, s_{h'+1}) \in [H - h] \times [S] \times [A] \times [S]\}$ such that $s_{h'+1} = s$ at $h' = h - 1$.

This update rule is based on the upper bound $\beta_h(s, a)$ from Lemma 1. If $\beta_h(s, a)$ is larger than the absolute difference between the step-wise Bellman update $U_h^k(s, a) = r_h^k(s, a) + V_{h+1}(s_{h+1})$ at the current episode k and $U_h^{k_0}(s, a)$, where k_0 is the last episode when (s, a, h) was visited, then the algorithm traces through the trajectory and increment Q-values by

$$\Delta Q_{h'}(s_{h'}, a_{h'}) = \text{sign}(U_h^k(s, a) - U_h^{k_0}(s, a))\gamma \frac{1}{H - h'} \beta_{h'}(s_{h'}, a_{h'}), \tag{3}$$

where sign returns the sign of an input, and γ is a scaling coefficient. We define $\gamma \stackrel{\text{def}}{=} \text{softmax}(Q_{h'}(s_{h'}, a_{h'}))$. This γ adaptively scales $\beta_{h'}(s_{h'}, a_{h'})$ based on the ratio of Q-values along the action dimension, ensuring the update amount associated with actions with a high Q-value is weighed more. The scaling coefficient $\frac{1}{H-h'}$ further scales down $\beta_{h'}(s_{h'}, a_{h'})$ based on its step index since state-action pairs in early steps branch out to different trajectories, and thus, get updated more often. We provide pseudocode in Appendix C.

This approach selectively updates specific $Q(s, a, h)$ positions, so it performs the best when the variance of $Q_h^m(s, \cdot)$ is small enough that a series of selective restarts can affect the actions chosen by the policy $\pi_h^m(s) = \arg\max_a Q_m^h(s, \cdot)$. Therefore, we may use as a base *any* algorithm that promises quick convergence in a *stationary* environment. Thus, for our empirical results in the following section, we utilize Wang et al. (2025)'s RANDOMIZEDQ. RANDOMIZEDQ is a Q-learning-based stationary RL algorithm that adopts the step-wise (*agile*) descents of Q-estimates, whereas RESTARTQ-UCB updates Q-table at every learning stage $\in \mathcal{L}$, in which the interval of the stages increases exponentially. Instead of UCB algorithms, RANDOMIZEDQ uses posterior sampling and Q-ensemble methods to address the exploration-exploitation trade-off. We discover that RANDOMIZEDQ functions significantly better with selective restarts than RESTARTQ-UCB does, due to the above characteristics. Therefore, in the following experiments, we focus on the performance of SelectiveRestarts + RANDOMIZEDQ.

The additional time complexity generated by SelectiveRestarts at each timestep is $\approx O(B)$, where $B \stackrel{\text{def}}{=} |\mathcal{T}_h(s, a)|$. Therefore, the total time complexity combined with the base RL algorithm is $\approx O(MH(C_\pi + B))$, where C_π is the time complexity of the base RL's policy update function. In practice, $\mathcal{T}_h(s, a)$ is reset at every learning stage, and visitations (h, s_h, a_h, s_{h+1}) do not proliferate much during a single stage, so B becomes significantly smaller than $H \times S \times A \times S$. The auxiliary space complexity is $\approx O(S(HA + H + A) + B)$.

5 Experimental setting

We demonstrate the efficacy of our approaches in two non-stationary MDP settings, RandomMDP and BDCL.

5.1 MDPs

Inspired by the need for robust testing on different environment types for non-stationary RL, we introduce the pseudorandomly generated environment RandomMDP. In brief, each action deterministically transitions a state to a randomly picked next state and rewards for each state, action combination are randomly generated to be between 0 and 1. To introduce non-stationarity, a new *target* MDP is randomly generated, and the available variation budget (decided by input parameters) is used to change the current MDP to the target. Once the target is reached, a new target is generated, and this process can repeat infinitely. Further detail of the generation process and input parameters can be found in Appendix D.

In this study, we also focus on BDCL, an episodic MDP designed to be particularly challenging for exploration. The environment presents two types of non-stationary dynamics based on configuration: *abrupt* and *gradual* variations. We explain the details of BDCL in Appendix E.

5.2 Methodology

We test our restart algorithms on RandomMDP with $A = 5$, $S = 5$, $H = 5$, $T = 50,000$ and varying input parameters to simulate different types of variation. We also test on gradual and abrupt BDCL $A = 5$, $H = 5$, $T = 100,000$, and fail probability 0.02. In the abrupt setting, changes happen at every $1,001$ episodes. We iterate each trial 5 times.

We test the following combinations of algorithms: (1) AdaptiveRestarts + PartialRestarts + RESTARTQ-UCB (ADAPARRESTARTQ-UCB) and (2) SelectiveRestarts + RANDOMIZEDQ (SELECTIVERANDOMIZEDQ). As baselines, we compare them with RESTARTQ-UCB, a random policy, and the optimal policy. Following the proof by Mao et al. (2022), b_Δ is removed (set to zero) from the update rule. In SELECTIVERANDOMIZEDQ, we set 20 ensembles, inflation coefficient $\kappa = 1$, and $n_0 = \frac{1}{4}$ prior transitions.

Since our main focus is the comparison against RESTARTQ-UCB, we performed hyperparameter tuning to find the best value for the probability hyperparameter δ. $\delta \in [0, 1]$ represents a probability required for the theoretical guarantees and impressive dynamic regret bound—the smaller δ, the more optimism is added but the more likely the bounds are to hold. However, in practice, δ may be as large as 2; then we set $\iota = \log\left(\frac{2}{\delta}\right)$ to zero on line 8 of Algorithm 1. Indeed, although this breaks the proofs in Mao et al. (2022), we find empirically that $\delta = 2$ is the optimal setting of this hyperparameter.

6 Results and discussion

Figure 3 shows the total reward of the algorithms on RandomMDP, abrupt, and gradual BDCL. Overall, ADAPARRESTARTQ-UCB and SELECTIVERANDOMIZEDQ achieve a significantly higher reward than RESTARTQ-UCB not only in RandomMDP but also in BDCL, in which an agent observes sparse reward signals and may struggle with exploration. These results demonstrate that our approaches successfully address the restart inefficiencies that we discuss in Section 3.1.

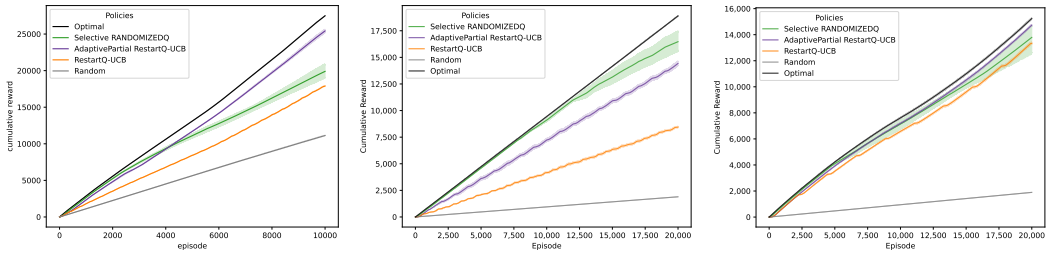

Figure 3: From the left, each plot corresponds to RandomMDP, abrupt BDCL, gradual BDCL. In RandomMDP, RESTARTQ-UCB with adaptive and partial restarts achieves near-optimal total reward, and shows great improvement over base RESTARTQ-UCB. In BDCL environments, RESTARTQ-UCB with adaptive and partial restarts, as well as SELECTIVERANDOMIZEDQ show an improved performance compared to base RESTARTQ-UCB. Notably, SELECTIVERANDOMIZEDQ has near-zero dynamic regret in abrupt BDCL at episode 7,500, showing the promise of this approach.

Partial and adaptive restarts Partial and adaptive restarts combined in RESTARTQ-UCB prove to be a substantial improvement over base RESTARTQ-UCB. Partial restarts function properly in environments regardless as to whether the non-stationarity stems from changes in reward, transition probabilities, or both. Adaptive restarts effectively handle both abrupt and gradual change types. Figure 3 shows that when combined in a BDCL environment with abrupt changes, adaptive partial restarts decrease the dynamic regret of RESTARTQ-UCB by 45%. In the RandomMDP, it decreases the dynamic regret by 74%, showing great improvement in restarts. We observe that the large difference in reward is caused by RESTARTQ-UCB restarting very often, usually at times when *not* restarting would still allow the agent to gain the maximum reward per episode. Adaptive restarts triggers very sparingly in comparison.

Selective restarts Selective restarts demonstrate near-optimal performance in the three environments, especially early in the runtime, significantly reducing dynamic regret. The most notable

example is shown in the middle of Figure 3. Selective restarts maintain almost optimal rewards in the first ten thousands of episodes, and similar trends are observable in RandomMDP and gradual BDCL in the first few thousands of episodes. In abrupt BDCL, selective restarts achieved 91% less dynamic regret relative to RESTARTQ-UCB. This is possible because of the algorithmic fit of selective restarts with agile stationary algorithms.

6.1 Limitations

The core limitation of this work is that our proposed methods are tested in only two settings and does not guarantee the performance of our approaches in every MDP. However, we believe that this work is an insightful step towards that for both theorists and practitioners. Below, we discuss in more depth how our approaches could be improved and how they may perform in environments that we did not examine.

Partial restarts Knowledge of Δ_p and Δ_r is a strong assumption in practice, and in many real-world scenarios this value is unknown. When implementing partial restarts without knowledge of variation budget, it must either be estimated through repeated sampling (which will likely have poor theoretical performance), or the budgets can be entered as values much higher than expected, so that the true budget shouldn't exceed them. In practice, this will be much closer to full restarts than the tightly-bounded partial restarts shown here. Future work could investigate how both approaches manage in different settings.

Adaptive restarts While adaptive restarts function well in the RandomMDP and BDCL environments shown here, worst-case environments can be constructed where these adaptive restarts will perform worse than scheduled ones. When implementing them, it should be tested if adaptive restarts will trigger at expected timings, and likely a combination of spaced-out scheduled restarts and adaptive restarts will give good performance while maintaining some degree of asymptotic guarantee.

Selective restarts Although we propose selective restarts as a new restart framework, the derivation of the update amount found in Equation 3 is still heuristic and needs a proof-based foundation. Moreover, we discover that selective restarts tend to start accumulating dynamic regret when the episode reaches a certain count, so future work should develop a new version derived from theoretical analysis.

7 Conclusion

In this paper, we focus on improving restart properties inspired by RESTARTQ-UCB. We propose partial, adaptive, and selective restarts, all of which successfully address the inherent problems of RESTARTQ-UCB (complete forgetting and inflexible restart timings) and shed light on achieving even quicker convergence in combination with other stationary algorithms. This work demonstrates that our approaches successfully elevate the efficacy of theoretically robust algorithms in experimental settings, shortening the gap between theory and practice that is prevalent in RL research.

Future work These results give rise to some promising future directions. Adaptive restarts can almost certainly be improved by taking a more theoretical approach to their design. It currently functions using a heuristic that works well in these environments, but it would not detect a change in a setting where the change does not decrease the agent's rate of reward. One notable finding in selective restarts is that SELECTIVERANDOMIZEDQ starts diverging from the optimal policy after some thousands of episodes and widens the confidence interval. Future improvements should focus on making its performance more robust by revisiting the update trigger condition and deriving a ΔQ that guarantees asymptotic performance (Equation 3). Nonetheless, the near-optimal performance early in the horizon implies positive possibilities for future improvements. Additionally, the asymptotic performance of these approaches could be found, and modifications may need to be made for more robust performance in general non-stationary MDPs. One can also apply our *restart wrappers* to other stationary algorithms. Since our approaches offer only simple, outer modifications to the algorithms, we believe this is a hopeful direction. Another direction is to integrate a future well-tailored stationary algorithm with these new restart frameworks to achieve potentially lower dynamic regret.

Acknowledgments

Aviva Prins was supported by Red Cell Partners. We would like to thank the students of REU-CAAR, along with Dr. William Gasarch, for their help and support during our research.

References

W. Mao, K. Zhang, R. Zhu, D. Simchi-Levi, and T. Başar, "Model-free non-stationary rl: Near-optimal regret and applications in multi-agent rl and inventory control," 2022. [Online]. Available: https://arxiv.org/abs/2010.03161

H. Wang, X. Xu, and Y. Chi, "Provably efficient and agile randomized q-learning," 2025. [Online]. Available: https://arxiv.org/abs/2506.24005

R. Gautron, O.-A. Maillard, P. Preux, M. Corbeels, and R. Sabbadin, "Reinforcement learning for crop management support: Review, prospects and challenges," *Comput. Electron. Agric.*, vol. 200, no. C, Sep. 2022. [Online]. Available: https://doi.org/10.1016/j.compag.2022.107182

T. Lu and A. Prins, "An online optimization-based decision support tool for small farmers in india: Learning in non-stationary environments," 2023. [Online]. Available: https://arxiv.org/abs/2311.17277

A. Mahajan, S. Hegde, E. Shay, D. Wu, and A. Prins, "Comparative analysis of multi-agent reinforcement learning policies for crop planning decision support," 2025. [Online]. Available: https://arxiv.org/abs/2412.02057

R. Tao, P. Zhao, J. Wu, N. F. Martin, M. T. Harrison, C. Ferreira, Z. Kalantari, and N. Hovakimyan, "Optimizing crop management with reinforcement learning and imitation learning," 2023. [Online]. Available: https://arxiv.org/abs/2209.09991

D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, pp. 484–503, 2016. [Online]. Available: http://www.nature.com/nature/journal/v529/n7587/full/nature16961.html

D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis, "Mastering chess and shogi by self-play with a general reinforcement learning algorithm," 2017. [Online]. Available: https://arxiv.org/abs/1712.01815

A. Agarwal, S. Kakade, M. Henaff, and W. Sun, "Pc-pg: policy cover directed exploration for provable policy gradient learning," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, ser. NIPS '20. Red Hook, NY, USA: Curran Associates Inc., 2020.

C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3, pp. 279–292, May 1992. [Online]. Available: https://doi.org/10.1007/BF00992698

V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," 2013. [Online]. Available: https://arxiv.org/abs/1312.5602

R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learn.*, vol. 8, no. 3–4, p. 229–256, May 1992. [Online]. Available: https://doi.org/10.1007/BF00992696

V. Konda and J. Tsitsiklis, "Actor-critic algorithms," in *Advances in Neural Information Processing Systems*, S. Solla, T. Leen, and K. Müller, Eds., vol. 12. MIT Press, 1999. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/1999/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf

B. C. da Silva, E. W. Basso, A. L. C. Bazzan, and P. M. Engel, "Dealing with non-stationary environments using context detection," in *Proceedings of the 23rd International Conference on Machine Learning*, ser. ICML '06. New York, NY, USA: Association for Computing Machinery, 2006, p. 217–224. [Online]. Available: https://doi.org/10.1145/1143844.1143872

H. Zhou, J. Chen, L. R. Varshney, and A. Jagmohan, "Nonstationary reinforcement learning with linear function approximation," 2024. [Online]. Available: https://arxiv.org/abs/2010.04244

P. Gajane, R. Ortner, and P. Auer, "Variational regret bounds for reinforcement learning," 2019. [Online]. Available: https://arxiv.org/abs/1905.05857

W. C. Cheung, D. Simchi-Levi, and R. Zhu, "Reinforcement learning for non-stationary markov decision processes: The blessing of (more) optimism," 2020. [Online]. Available: https://arxiv.org/abs/2006.14389

C. Jin, Z. Allen-Zhu, S. Bubeck, and M. I. Jordan, "Is q-learning provably efficient?" in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31. Curran Associates, Inc., 2018. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2018/file/d3b1fb02964aa64e257f9f26a31f72cf-Paper.pdf

Z. Zhang, Y. Zhou, and X. Ji, "Almost optimal model-free reinforcement learning via reference-advantage decomposition," 2020. [Online]. Available: https://arxiv.org/abs/2004.10019

Appendix A: Proof of Lemma 1

Lemma 1. *For any triple (s, a, h) and any episodes $k_1, k_2 \in [K]$, it holds that*

$$\left|Q_h^{k_2,\star}(s,a) - Q_h^{k_1,\star}(s,a)\right| \leq \Delta_r + \frac{1}{2}\Delta_p \min_{k\in k_1,k_2}\left[\max_{s,a,h'>h}\left[Q_{h'}^{k,\star}(s,a)\right]\right].$$

Proof. First, let us revise some notation. Let $Q_h^{k,\star}(s,a)$ be the Q-score of state s and action a given the optimal policy $\pi^\star$ at episode k and timestep h. Define $V_h^{k,\star}(s,a)$ similarly. Define $\mathbb{E}\left[V_{h+1}^{k,\star}(s)\right] := \sum_{s'\in S}\left[P_h^k(s'|s,a)V_{h+1}^{k,\star}(s')\right]$. Let Δ_r and Δ_p be the total reward and probability variation budgets, respectively. Let $\Delta_{r,h}$ and $\Delta_{p,h}$ be the variation budgets confined to a single timestep, such that

$$\sum_{h=1}^{H}\Delta_{r,h} = \Delta_r\,,\ \sum_{h=1}^{H}\Delta_{p,h} = \Delta_p.$$

Finally, let $\alpha_h = \sum_{h'=h}^{H}\left[\Delta_{r,h'} + \frac{1}{2}\Delta_{p,h'}\min_{k\in k_1,k_2}\left[\max_{s,a}Q_{h'+1}^{k,\star}(s,a)\right]\right]$.

Our proof will be via backwards induction on h. This generalizes to all epochs, so d will be omitted in notation. Without loss of generality, assume $k_2 > k_1$. If this is not the case, flipping the order of sums from k_1 to k_2 will still hold. Lastly, if $k_2 = k_1$, the proof is trivial.

Base case: $(h = H)$

$$\begin{aligned}
\left|Q_H^{k_2,\star}(s,a) - Q_H^{k_1,\star}(s,a)\right| &= \left|r_H^{k_2}(s,a) + \mathbb{E}\left[V_{H+1}^{k_2,\star}(s')\right] - \ r_H^{k_1}(s,a) - \mathbb{E}\left[V_{H+1}^{k_1,\star}(s')\right]\right| \\
&= \left|r_H^{k_2}(s,a) - r_H^{k_1}(s,a)\right| \\
&\leq \sum_{k=k_1}^{k_2-1}\left|r_H^{k+1}(s,a) - r_H^{k}(s,a)\right| \\
&\leq \sum_{k=1}^{K-1}\left|r_H^{k+1}(s,a) - r_H^{k}(s,a)\right| \\
&\leq \Delta_{r,H}
\end{aligned}$$

Inductive Hypothesis:

$$Q_{h+1}^{k_2,\star}(s,a) - Q_{h+1}^{k_1,\star}(s,a) \leq \alpha_{h+1}$$

Inductive Step:

$$
\begin{aligned}
Q_h^{k_2,\star}(s,a) - Q_h^{k_1,\star}(s,a) &= r_h^{k_2}(s,a) - \ r_h^{k_1}(s,a) + \mathbb{E}\left[V_{h+1}^{k_2,\star}(s')\right] - \mathbb{E}\left[V_{h+1}^{k_1,\star}(s')\right] \\
&\leq \sum_{k=k_1}^{k_2-1} \left[r_h^{k+1}(s,a) - r_h^k(s,a)\right] + \mathbb{E}\left[V_{h+1}^{k_2,\star}(s')\right] - \mathbb{E}\left[V_{h+1}^{k_1,\star}(s')\right] \\
&\leq \sum_{k=1}^{K-1} \left[r_h^{k+1}(s,a) - r_h^k(s,a)\right] + \mathbb{E}\left[V_{h+1}^{k_2,\star}(s')\right] - \mathbb{E}\left[V_{h+1}^{k_1,\star}(s')\right] \\
&\leq \Delta_{r,h} + \mathbb{E}\left[V_{h+1}^{k_2,\star}(s')\right] - \mathbb{E}\left[V_{h+1}^{k_1,\star}(s')\right] \\
&= \Delta_{r,h} + \sum_{s'\in S} P_h^{k_2}(s'|s,a)V_{h+1}^{k_2,\star}(s') - P_h^{k_1}(s'|s,a)V_{h+1}^{k_1,\star}(s') \\
&= \Delta_{r,h} + \sum_{s'\in S} P_h^{k_2}(s'|s,a)Q_{h+1}^{k_2,\star}(s',\pi_{h+1}^{k_2,\star}(s')) - P_h^{k_1}(s'|s,a)V_{h+1}^{k_1,\star}(s') \\
&= \Delta_{r,h} + \sum_{s'\in S} P_h^{k_2}(s'|s,a)Q_{h+1}^{k_2,\star}(s',\pi_{h+1}^{k_2,\star}(s')) - P_h^{k_1}(s'|s,a)Q_{h+1}^{k_1,\star}(s',\pi_{h+1}^{k_1,\star}(s')) \\
&\leq \Delta_{r,h} + \sum_{s'\in S} P_h^{k_2}(s'|s,a)Q_{h+1}^{k_2,\star}(s',\pi_{h+1}^{k_2,\star}(s')) - P_h^{k_1}(s'|s,a)Q_{h+1}^{k_1,\star}(s',\pi_{h+1}^{k_2,\star}(s')) \\
&\quad \textbf{By the I.H.} \\
&\leq \Delta_{r,h} + \sum_{s'\in S} P_h^{k_2}(s'|s,a)(Q_{h+1}^{k_1,\star}(s',\pi_{h+1}^{k_2,\star}(s')) + \alpha_{h+1}) - P_h^{k_1}(s'|s,a)Q_{h+1}^{k_1,\star}(s',\pi_{h+1}^{k_2,\star}(s')) \\
&= \Delta_{r,h} + \sum_{s'\in S} (P_h^{k_2}(s'|s,a) - P_h^{k_1}(s'|s,a))Q_{h+1}^{k_1,\star}(s',\pi_{h+1}^{k_2,\star}(s')) + P_h^{k_2}(s'|s,a)\alpha_{h+1} \\
&= \Delta_{r,h} + \alpha_{h+1} + \sum_{s'\in S} (P_h^{k_2}(s'|s,a) - P_h^{k_1}(s'|s,a))Q_{h+1}^{k_1,\star}(s',\pi_{h+1}^{k_2,\star}(s'))
\end{aligned}
$$

We can find this additional result by applying the I.H. to the right instead of the left side of the sum:

$$
\begin{aligned}
&\Delta_{r,h} + \sum_{s'\in S} P_h^{k_2}(s'|s,a)Q_{h+1}^{k_2,\star}(s',\pi_{h+1}^{k_2,\star}(s')) - P_h^{k_1}(s'|s,a)Q_{h+1}^{k_1,\star}(s',\pi_{h+1}^{k_2,\star}(s')) \\
&\quad \textbf{By the I.H.} \\
&\leq \Delta_{r,h} + \sum_{s'\in S} P_h^{k_2}(s'|s,a)Q_{h+1}^{k_2,\star}(s',\pi_{h+1}^{k_2,\star}(s')) - P_h^{k_1}(s'|s,a)(Q_{h+1}^{k_2,\star}(s',\pi_{h+1}^{k_2,\star}(s')) - \alpha_{h+1}) \\
&= \Delta_{r,h} + \sum_{s'\in S} (P_h^{k_2}(s'|s,a) - P_h^{k_1}(s'|s,a))Q_{h+1}^{k_2,\star}(s',\pi_{h+1}^{k_2,\star}(s')) + P_h^{k_1}(s'|s,a)\alpha_{h+1} \\
&= \Delta_{r,h} + \alpha_{h+1} + \sum_{s'\in S} (P_h^{k_2}(s'|s,a) - P_h^{k_1}(s'|s,a))Q_{h+1}^{k_2,\star}(s',\pi_{h+1}^{k_2,\star}(s'))
\end{aligned}
$$

We now bound $\sum_{s'\in S}\Big[(P_h^{k_2}(s'|s,a) - P_h^{k_1}(s'|s,a))Q_{h+1}^{k_1,\star}(s',\pi_{h+1}^{k_2,\star}(s'))\Big]$. (We want to show $\leq \frac{1}{2}\Delta_{p,h}\max_{s,a} Q_{h+1}^{k_1,\star}(s,a)$).

Note: The following steps are also used to bound:

$\sum_{s'\in S}\Big[(P_h^{k_2}(s'|s,a) - P_h^{k_1}(s'|s,a))Q_{h+1}^{k_2,\star}(s',\pi_{h+1}^{k_2,\star}(s'))\Big]$.

Because $P_h^{k_2}(\cdot|s,a)$ and $P_h^{k_1}(\cdot|s,a)$ are probability vectors, the following is true:

$$
\sum_{s'\in S} P_h^{k_2}(s'|s,a) = 1 = \sum_{s'\in S} P_h^{k_1}(s'|s,a)
$$

$$\rightarrow \sum_{s' \in S} \left[P_h^{k_2}(s'|s,a) - P_h^{k_1}(s'|s,a) \right] = 0$$

By the definition of $\Delta_{p,h}$, the following is also true:

$$\sum_{s' \in S} \left[\left| P_h^{k_2}(s'|s,a) - P_h^{k_1}(s'|s,a) \right| \right] \leq \Delta_{p,h}$$

For ease of notation going forward, let $\theta_i = P_h^{k_2}(s_i|s,a) - P_h^{k_1}(s_i|s,a)$, $Q_i = Q_{h+1}^{k_1,\star}(s_i, \pi_{h+1}^{k_2,\star})$, and let $n = |S|$, the number of states. Therefore, the above equations and inequality become:

$$\sum_{i=1}^{n} \left| \theta_i \right| \leq \Delta_{p,h}$$
$$\sum_{i=1}^{n} \theta_i = 0$$

$$\sum_{s' \in S} \left[(P_h^{k_2}(s'|s,a) - P_h^{k_1}(s'|s,a)) Q_{h+1}^{k_1,\star}(s', \pi_{h+1}^{k_2,\star}) \right] = \sum_{i=1}^{n} \theta_i Q_i$$

We will show the sum of all positive θ_i is less than or equal to $\frac{1}{2}\Delta_{p,h}$. Since the sum of the elements in θ_i is equal to 0,

$$\sum_{i:\theta_i > 0} \theta_i = - \sum_{i:\theta_i < 0} \theta_i$$

Therefore,

$$\sum_{i=1}^{n} |\theta_i| = \sum_{i:\theta_i > 0} \theta_i - \sum_{i:\theta_i < 0} \theta_i = 2 \sum_{i:\theta_i > 0} \theta_i$$

Since the sum $\sum_{i=1}^{n} |\theta_i|$ is bounded by $\Delta_{p,h}$,

$$2 \sum_{i:\theta_i > 0} \theta_i \leq \Delta_{p,h}$$
$$\sum_{i:\theta_i > 0} \theta_i \leq \frac{1}{2} \Delta_{p,h}$$

Now we return to our sum $\sum_{i=1}^{n} \theta_i Q_i$.

Since all $Q_i \geq 0$:

$$\begin{aligned}
\sum_{s'\in S}\Big[(P_h^{k_2}(s'|s,a)-P_h^{k_1}(s'|s,a))Q_{h+1}^{k_1,\star}(s',\pi_{h+1}^{k_2,\star})\Big] &= \sum_{i=1}^{n}\theta_i Q_i \\
&\leq \sum_{i=1:\theta_i>0}^{n}\theta_i Q_i \\
&\leq \sum_{i=1:\theta_i>0}^{n}\theta_i \max_s(Q_s) \\
&\leq \frac{1}{2}\Delta_{p,h}\max_s(Q_s) \\
&= \frac{1}{2}\Delta_{p,h}\max_s(Q_{h+1}^{k_1,\star}(s,\pi_{h+1}^{k_2,\star}(s))) \\
&\leq \frac{1}{2}\Delta_{p,h}\max_{s,a}(Q_{h+1}^{k_1,\star}(s,a))
\end{aligned}$$

Similarly,

$$\sum_{s'\in S}\Big[(P_h^{k_2}(s'|s,a)-P_h^{k_1}(s'|s,a))Q_{h+1}^{k_2,\star}(s',\pi_{h+1}^{k_2,\star})\Big]\leq \frac{1}{2}\Delta_{p,h}\max_{s,a}(Q_{h+1}^{k_2,\star}(s,a))$$

Combining all above steps,

$$\begin{aligned}
Q_h^{k_2,\star}(s,a)-Q_h^{k_1,\star}(s,a) &\leq \Delta_{r,h}+\alpha_{h+1}+\sum_{s'\in S}\left[(P_h^{k_2}(s'|s,a)-P_h^{k_1}(s'|s,a))Q_{h+1}^{k_1,\star}(s',\pi_{h+1}^{k_2,\star})\right] \\
&\leq \Delta_{r,h}+\alpha_{h+1}+\frac{1}{2}\Delta_{p,h}\max_{s,a}(Q_{h+1}^{k_1,\star}(s,a)) \\
&= \Delta_{r,h}+\sum_{h'=h+1}^{H}\left[\Delta_{r,h'}+\frac{1}{2}\Delta_{p,h'}\max_{s,a}Q_{h'+1}^{k_1,\star}(s,a)\right]+\frac{1}{2}\Delta_{p,h}\max_{s,a}(Q_{h+1}^{k_1,\star}(s,a)) \\
&= \sum_{h'=h}^{H}\left[\Delta_{r,h'}+\frac{1}{2}\Delta_{p,h'}\max_{s,a}Q_{h'+1}^{k_1,\star}(s,a)\right]
\end{aligned}$$

Through a similar process, it can be shown that:

$$Q_h^{k_2,\star}(s,a)-Q_h^{k_1,\star}(s,a)\leq \sum_{h'=h}^{H}\left[\Delta_{r,h'}+\frac{1}{2}\Delta_{p,h'}\max_{s,a}Q_{h'+1}^{k_2,\star}(s,a)\right]$$

Combining the two results:

$$\begin{aligned}
Q_h^{k_2,\star}(s,a)-Q_h^{k_1,\star}(s,a) &\leq \sum_{h'=h}^{H}\left[\Delta_{r,h'}+\frac{1}{2}\Delta_{p,h'}\min_{k\in k_1,k_2}\left[\max_{s,a}Q_{h'+1}^{k,\star}(s,a)\right]\right] \\
&= \alpha_h \\
&\leq \Delta_r+\frac{1}{2}\Delta_p\min_{k\in k_1,k_2}\left[\max_{s,a,h'>h}Q_{h'}^{k,\star}(s,a)\right]
\end{aligned}$$

Finally, because of our assumption (without loss of generality) that $k_2 > k_1$, bounding $Q_h^{k_1,\star}(s,a) - Q_h^{k_2,\star}(s,a)$ will yield the same result. Therefore:

$$\left|Q_h^{k_2,\star}(s,a) - Q_h^{k_1,\star}(s,a)\right| \leq \Delta_r + \frac{1}{2}\Delta_p \min_{k\in k_1,k_2}\left[\max_{s,a,h'>h} Q_{h'}^{k,\star}(s,a)\right]$$

□

Appendix B: AdaptiveRestarts Algorithm

The following algorithm is RESTARTQ-UCB modified to use adaptive restarts, described in 4.2. For clarity, all variables using `snake_case` are newly introduced for adaptive restarts.

Algorithm 2: RESTARTQ-UCB (Hoeffding) with Adaptive Restarts

while `restart` $==$ *`True`* **do**
  Initialize: $V_h(s) \leftarrow H-h+1$, $Q_h(s,a) \leftarrow H-h+1$, $N_h(s,a) \leftarrow 0$, $\check{N}_h(s,a) \leftarrow 0$, $\check{r}_h(s,a) \leftarrow 0$, $\check{v}_h(s,a) \leftarrow 0$, $\check{\mu}_h(s,a) \leftarrow 0$, $\check{\sigma}_h(s,a) \leftarrow 0$, $\mu_h^{\text{ref}}(s,a) \leftarrow 0$, $\sigma_h^{\text{ref}}(s,a) \leftarrow 0$, $V_h^{\text{ref}}(s) \leftarrow H$, for all $(s,a,h) \in \mathcal{S}\times\mathcal{A}\times[H]$.
  Initialize (restart vars): `W` $\leftarrow 0$, $r_L \leftarrow 0$, $r_B \leftarrow 0$, $r_C \leftarrow 0$, `trueCount` $\leftarrow 0$, `nonCount` $\leftarrow 0$, `lastReset` $\leftarrow 0$. `rewardHistory` $\leftarrow [\,]$.
  for *episode* $k \leftarrow k+1$ *to* T **do**
    observe s_1, ;
    `episodeReward` $\leftarrow 0$;
    for *step* $h \leftarrow 1$ *to* H **do**
      Take action $a_h \leftarrow \arg\max_a Q_h(s_h,a)$; **Receive** reward $R_h(s_h,a_h)$ and observe s_{h+1};
      $\check{r}_h(s_h,a_h) \leftarrow \check{r}_h(s_h,a_h) + R_h(s_h,a_h)$, $\check{v}_h(s_h,a_h) \leftarrow \check{v}_h(s_h,a_h) + V_{h+1}(s_{h+1})$, `episodeReward` $\leftarrow$ `episodeReward` $+ R_h(s_h,a_h)$;
      $N_h(s_h,a_h) \leftarrow N_h(s_h,a_h)+1$, $\check{N}_h(s_h,a_h) \leftarrow \check{N}_h(s_h,a_h)+1$;
      if $N_h(s_h,a_h) \in \mathcal{L}$ **then**
        $b_h \leftarrow \sqrt{\frac{H^2}{N_h(s_h,a_h)}\,\iota} + \sqrt{\frac{1}{N_h(s_h,a_h)}\,\iota}$, $b_\Delta \leftarrow \Delta_r^{(d)} + H\cdot\Delta_p^{(d)}$;
        $Q_h(s_h,a_h) \leftarrow \min\left\{\frac{\check{r}_h(s_h,a_h)}{\check{N}_h(s_h,a_h)} + \frac{\check{v}_h(s_h,a_h)}{\check{N}_h(s_h,a_h)} + b_h + 2b_\Delta, Q_h(s_h,a_h)\right\}$;
        $V_h(s_h) \leftarrow \max_a Q_h(s_h,a)$;
        if *optimal action at* (s_h,h) *unchanged* **then**
          `nonCount` $\leftarrow$ `nonCount` $+1$;
        else
          `trueCount` $\leftarrow$ `trueCount` $+1$;
        if `trueCount` $\geq H^2$ **then**
          `trueCount`, `nonCount` $\leftarrow 0, 0$; `lastReset` $\leftarrow k$; ▷ still learning
        if `nonCount` $\geq H^2$ **then**
          $W \leftarrow k -$ `lastReset`; ▷ learning done
        $\check{N}_h(s_h,a_h) \leftarrow 0$, $\check{r}_h(s_h,a_h) \leftarrow 0$, $\check{v}_h(s_h,a_h) \leftarrow 0$;
    Append `episodeReward` to `rewardHistory`;
    if `W` > 0 **then**
      $r_L \leftarrow \sum_{j=1}^{W}$ `rewardHistory`$[j]$; ▷ learning reward
      $r_C \leftarrow \sum_{j=k-W+1}^{k}$ `rewardHistory`$[j]$; ▷ current reward
      $r_B \leftarrow \max(r_B, r_C)$; ▷ best reward so far
      $R_{\text{no}} \leftarrow r_C \cdot \frac{T-t}{HW}$, ;
      $R_{\text{yes}} \leftarrow r_L + r_B \cdot \left(\frac{T-t}{HW} - 1\right)$;
      if $R_{no} < R_{yes}$ **then**
        Restart: reinitialize all variables as at start of epoch.

Appendix C: SelectiveRestarts Algorithm

The following is the algorithm of our SelectiveRestarts. In the following pseudocode, only trajectory $\mathcal{T}$ and the trajectory dictionary T are initialized for SelectiveRestarts. We used a dictionary to store the trajectory, which significantly reduces the space complexity over using an $H \times S \times A \times S$ matrix, where the dimensions correspond to (h, s_h, a_h, s_{h+1}). The time complexity of the function $f_{\mathcal{T}}$ that retrieves a set of $(h', s_{h'}, a_{h'}, s_{h'+1})$ quadruples from T may vary, but in our algorithm, it is $O(B)$.

Algorithm 3: SelectiveRestarts

Input : Learning stages: $\mathcal{L}$; Visitation count: $N_h(s,a)$; Set of arbitrary visitation counts: $\hat{N}_h(s,a) = \{\hat{N}_{1,h}(s,a), \hat{N}_{2,h}(s,a), \ldots, \hat{N}_{n,h}(s,a)\}$

Require : Policy update function: f_π; Value update function: f_V; Trajectory calculation function: $f_{\mathcal{T}}$; Current undiscounted step-wise Bellman update: $U_h^k(s,a)$; Bellman update at past episode k_0: $U_h^{k_0}(s,a)$; Upper bound of $\left|Q_h^{k_1,\star}(s,a) - Q_h^{k_2,\star}(s,a)\right|$ for $0 \le k_1 < k_2 \le K$: $\beta_h(s,a)$; Standard deviation of Q along actions: $\sigma(Q_h(s,\cdot))$;

Initialize : Trajectory $\mathcal{T}_h(s,a) \leftarrow \phi$; Trajectory dictionary: T

for *episode* $m \leftarrow 1$ **to** M **do**
  Observe the initial state s_1^k;
  for *step* $h \leftarrow 1$ **to** H **do**
    Take action $a_h^m = \pi_h^m(s_h^m)$; Observe s_{h+1}^m; Receive reward $r_h^m(s_h^m, a_h^m)$
    $T(h, s_h^m, a_h^m, s_{h+1}^m) \leftarrow 1$
    `// Execute the base policy update`
    $f_\pi(s_h^m, a_h^m, s_{h+1}^m, r_h^m(s_h^m, a_h^m))$;
    `// Calculate undiscounted step-wise Bellman update`
    $U_h^m(s_h^m, a_h^m) = r_h^m(s_h^m, a_h^m) + V_{h+1}^m(s_{h+1}^m)$
    if $|U_h^{m_0}(s_h^m, a_h^m) - U_h^m(s_h^m, a_h^m)| \ge \beta_h(s_h^m, a_h^m)$ **then**
      `// Find the trajectory to` (h, s_h^m, a_h^m)
      $\mathcal{T}_h(s_h^m, a_h^m) \leftarrow f_{\mathcal{T}}(T, h, s_h^m, a_h^m)$
      for $(h', s_{h'}, a_{h'}, s_{h'+1}) \in \mathcal{T}_h(s_h^m, a_h^m)$ **do**
        `// Calculate update amount`
        $\Delta Q_{h'}(s_{h'}, a_{h'}) = \text{sign}(U_h^m(s,a) - U_h^{m_0}(s,a))\gamma \frac{1}{H-h'} \beta_{h'}(s_{h'}, a_{h'})$
        $Q_{h'}(s_{h'}, a_{h'}) \leftarrow Q_{h'}(s_{h'}, a_{h'}) + \Delta Q_{h'}(s_{h'}, a_{h'})$
        $V_{h'}(s_{h'}) \leftarrow f_V(Q)$
      `// If any, reset other visitation counts`
      $\hat{N}_h(s_h^m, a_h^m) \leftarrow \{0\}^n$
      $U_h^{m_0}(s_h^m, a_h^m) \leftarrow U_h^m(s_h^m, a_h^m)$
    if $N_h(s_h^k, a_h^k) \in \mathcal{L}$ **then**
      `// Reset the trajectory table`
      for $(h', s_{h'}, a_{h'}, s_{h'+1}) \in \mathcal{T}_h(s_h^m, a_h^m)$ **do**
        $T(h', s_{h'}, a_{h'}, s_{h'+1}) \leftarrow 0$

Appendix D: RandomMDP

RandomMDP has many input parameters that influence generation. First, to define the state-action space and random seed, it requires `N_STATES`, `N_ACTIONS`, `EPISODE_LENGTH`, and `MDP_SEED`. To determine the total variation budgets, Δ_r and Δ_p, it takes in `TOTAL_DELTA_R` and `TOTAL_DELTA_P` respectively. To determine how abrupt each kind of variation is, there is `DELTA_R_ABRUPTNESS` and `DELTA_P_ABRUPTNESS`, where each specifies the percent of episodes with 0 Δ_r and Δ_p. `DELTA_R_BUDGET_DISTRIBUTION` and `DELTA_P_BUDGET_DISTRIBUTION` describe how the total budget is divided between episodes that have non-zero variation. "uniform" has every episode have the same amount of variation, while "linear" linearly increases the variation per episode starting from 0. `FAIL_PROBABILITY` is the chance of transitioning to a random, non-target state. Lastly, `REWARD_SPARSITY` is the percent of "low-reward" state-actions, where reward is sampled from

[0, 0.2] instead of [0,1]. Below is the generation procedure for RandomMDP given these input parameters.

In the simulations shown in this paper, RandomMDP was run with the following parameters: `EPISODE_LENGTH: 5, N_STATES: 5, N_ACTIONS: 5, TOTAL_DELTA_R: 5, TOTAL_DELTA_P: 10, DELTA_R_ABRUPTNESS: 0.999, DELTA_P_ABRUPTNESS: 0.5, DELTA_P_BUDGET_DISTRIBUTION: uniform, DELTA_R_BUDGET_DISTRIBUTION: uniform, FAIL_PROBABILITY: 0.05, REWARD_SPARSITY: 0.8`

Algorithm 4: RandomMDP generation

`// First generate the per-episode variation budget`

1 $\texttt{num\_of\_eps\_with\_}\Delta_\texttt{r} \leftarrow K(1 - \texttt{DELTA\_R\_ABRUPTNESS})$

2 $\texttt{avg\_}\Delta_\texttt{r}\texttt{\_per\_ep} \leftarrow \frac{\texttt{TOTAL\_DELTA\_R}}{\texttt{num\_of\_eps\_with\_}\Delta_\texttt{r}}$

3 Define Δ_r^i as the reward variation budget for episode i.

4 **for** *episode* $i \leftarrow 1$ **to** $\texttt{num\_of\_eps\_with\_}\Delta_\texttt{r}$ **do**

5 Assign Δ_r^i based on the budget distribution defined in the config. If uniform, each gets the same. If linear, variation budget for each increases from 0 to $2(\texttt{avg\_}\Delta_\texttt{r}\texttt{\_per\_ep})$.

6 **for** *episode* $i \leftarrow \texttt{num\_of\_eps\_with\_}\Delta_\texttt{r}$ **to** K **do**

7 $\Delta_r^i \leftarrow 0$

8 Based on `MDP_SEED`, randomly swap each Δ_r^i with another Δ_r^i, so that the budget is distributed across all episodes.

9 Lines 1-9 using Δ_p in place of Δ_r.

10 Generate `N_STATES` states and `N_ACTIONS` actions.

11 **for** *state* $s \leftarrow s_0$ **to** s_S **do**

12 **for** *action* $a \leftarrow a_0$ **to** a_A **do**

13 **for** *timestep* $h \leftarrow 1$ **to** H **do**

14 $P_h^1(s'|s,a) \leftarrow 1 - \texttt{FAIL\_PROBABILITY}$, where s' is randomly sampled from S.

15 $P_h^1((\forall s \neq s'|s,a) \leftarrow \frac{\texttt{FAIL\_PROBABILITY}}{\texttt{N\_STATES}-1}$

16 $r_h^1(s,a) \leftarrow$ random value $\in [0,1]$, for $1 - \texttt{REWARD\_SPARSITY}$ % of (s,a,h) triples.

17 $r_h^1(s,a) \leftarrow$ random value $\in [0,0.2]$, for `REWARD_SPARSITY` % of (s,a,h) triples.

18 P^1 and r^1 are the `current_P` and `current_r` of the MDP.

19 Repeat lines 9-17 to generate `target_P` and `target_r`

20 $needed\Delta_r,\ needed\Delta_p \leftarrow$ total Δ_r and Δ_p to go from the current to the target r and P functions, respectively.

21 $\alpha_r,\ \alpha_p \leftarrow 0,\ 0$

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

23 $\alpha_r \mathrel{+}= \frac{\texttt{needed}\Delta_\texttt{r}}{\Delta_r^{k-1}}$

24 $\alpha_p \mathrel{+}= \frac{\texttt{needed}\Delta_\texttt{p}}{\Delta_p^{k-1}}$

25 **if** $\alpha_r > 1$ **then**

26 $\texttt{current\_r} \leftarrow \texttt{target\_r}$

27 Generate new `target_r` using the same process as lines 16 and 17.

28 $\alpha_r \leftarrow 0$

29 **if** $\alpha_p > 1$ **then**

30 $\texttt{current\_P} \leftarrow \texttt{target\_P}$

31 Generate new `target_P` using the same process as lines 14 and 15.

32 $\alpha_p \leftarrow 0$

33 $\forall(h,s,a,s')P_h^k(s'|s,a) \leftarrow \texttt{current\_P}_h(s'|s,a)(1-\alpha_p) + \texttt{target\_P}_h(s'|s,a)(\alpha_p)$

34 $\forall(h,s,a)r_h^k(s,a) \leftarrow \texttt{current\_r}_h(s,a)(1-\alpha_p) + \texttt{target\_r}_h(s,a)(\alpha_p)$

Appendix E: Bidirectional Diabolical Combination Locks (BDCL)

BDCL starts with a fixed initial state and transitions to either *lock1* or *lock2* depending on an action. Once the agent reaches one of the locks, it needs to keep choosing correct actions H times to stay in the lock and receive a final reward: $r_H(s_{\text{lock}}, a_{\text{correct}}) \in \{0.25, 1.0\}$, depending on which lock the agent is in. The locks return reward 0 until the final reward. If the agent chooses an incorrect action, it goes to the *sink* state, in which the agent can only receive a slight reward and cannot go back to a lock, no matter what action it takes. There are *abrupt* and *gradual* settings. In the abrupt setting, the final rewards of lock1 and lock2 switch at every some number of episodes. *Fail probability* directs the agent to the sink from a lock with probability p when it takes a correct action. In gradual BDCL, the transition probability from the initial state to a lock given a particular action changes at a linear scale at every episode, throughout the horizon.