# Non-Stationarity in RL Environments

*Ali Najar*[1], *Mazdak Teymourian*[1]
[1]*Department of Computer Engineering, Sharif University of Technology, Tehran, Iran*
[1]*{anajar13750,mazdak.tey}@gmail.com*

## Abstract

Reinforcement learning (RL) agents traditionally assume a stationary environment, yet many real-world settings-from robotics to finance-exhibit evolving dynamics and reward structures that violate this assumption. This non-stationarity manifests as drifting transition probabilities, shifting objectives, or co-learning agents, leading to outdated value estimates, policy instability, and performance degradation. Motivated by the ubiquity of time-varying phenomena in practical applications, our work investigates algorithms that provably adapt to changing MDPs without prior knowledge of drift magnitude or change points. We aim to (1) characterize the fundamental limits of learning under bounded and unbounded non-stationarity, (2) design both model-based and model-free methods, leveraging sliding-window estimation, exponential forgetting, and optimism-driven exploration, to minimize dynamic regret, and (3) extend these techniques to deep and multi-agent settings where high-dimensional representations and adversarial co-learners exacerbate non-stationarity. Through theoretical analysis and empirical evaluation, we seek to deliver scalable RL solutions that maintain robust performance across a spectrum of evolving environments.

**Keywords: Non-stationary RL; non-stationary MDPs; policy optimization; dynamic regret.**

## Introduction

Reinforcement learning (RL) provides a framework for sequential decision-making where an agent interacts with an environment to maximize cumulative reward. This interaction is typically modeled as a Markov Decision Process (MDP), defined by a tuple

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, H),$$

where $\mathcal{S}$ is the state space, $\mathcal{A}$ the action space, $P(\cdot \mid s, a)$ the transition kernel, $r(s, a)$ the reward function, and $H$ the horizon. A **policy** $\pi(a \mid s)$ maps states to action distributions. The agent's objective in a stationary MDP is to find a policy maximizing the expected return:

$$V^{\pi}(s) = \mathbb{E}_{\pi}\left[\sum_{h=1}^{H} r(s_h, a_h) \,\middle|\, s_1 = s\right].$$

In stationary settings, both $P$ and $r$ are fixed over time, allowing the optimal policy $\pi^{\star}$ to be learned through repeated interaction.
Let time/episode $t \in [T]$. The environment is a time-varying MDP

$$\mathcal{M}_t = (\mathcal{S}, \mathcal{A}, P_t, r_t, H), \qquad P_t(\cdot \mid s, a),\ r_t(s, a) \text{ may depend on } t.$$

Two useful ways to formalize non-stationarity:

**(A) Bounded variation (drift budget).** Define per-step variation and budgets

$$B_r := \sum_{t=1}^{T-1} \max_{(s,a)} \left|r_{t+1}(s,a) - r_t(s,a)\right|, \qquad B_p := \sum_{t=1}^{T-1} \max_{(s,a)} \left\|P_{t+1}(\cdot \mid s,a) - P_t(\cdot \mid s,a)\right\|_1.$$

Piecewise-stationary is a special case with finitely many change points; then $B_r$, $B_p$ scale with the number/magnitude of changes.

**(B) Lipschitz (smooth drift).** For all $t, \Delta \geq 1$ and $(s, a)$,

$$\left|r_{t+\Delta}(s,a) - r_t(s,a)\right| \leq L_r \Delta, \qquad W_1(P_{t+\Delta}(\cdot \mid s,a),\ P_t(\cdot \mid s,a)) \leq L_p \Delta,$$

with $W_1$ the 1-Wasserstein distance. (Set $\Delta = 1$ for per-step drift.)
**Goal: track the moving optimum.**
Let

$$\pi_t^{\star} \in \arg\max_{\pi} V_{t,1}^{\pi}(s_{t,1}), \qquad V_{t,1}^{\pi}(s) := \mathbb{E}_{\pi, P_t}\left[\sum_{h=1}^{H} r_t(s_h, a_h) \,\middle|\, s_1 = s\right].$$

Non-stationarity makes past data stale; algorithms must **adapt** (forget/discount old data) while **exploring** to control estimation error under $(B_r, B_p)$ or $(L_r, L_p)$.

## Risk-Averse Tree-Search

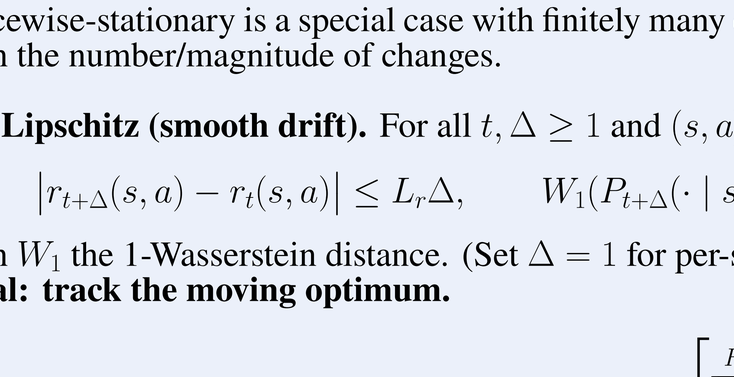



Figure 1: RATS [1] (Risk-Averse Tree-Search): a robust minimax planner for non-stationary MDPs that treats nature as an adversary and picks the action maximizing worst-case return over all Lipschitz-consistent evolutions of rewards/transitions.

**Algorithm 1** RATS Algorithm

**procedure** RATS($s_0, t_0, D$)
  $\nu_0 \leftarrow$ RootNode($s_0, t_0$); Minimax($\nu_0, D$); **return** $\arg\max_{\nu' \in \nu_0.\text{children}} \nu'.\text{action}$
**end procedure**
**function** Minimax($\nu, D$)
  **if** $\nu$ is decision node **then**
    **return** $\nu.\text{value} \leftarrow \begin{cases} \text{HeuristicValue}(\nu.\text{state}), & \text{terminal or } \nu.\text{depth} = D \\ \max_{\nu' \in \nu.\text{children}} \text{Minimax}(\nu', D), & \text{otherwise} \end{cases}$
  **else**
    **return** $\nu.\text{value} \leftarrow \min_{(p,R) \in \Delta_{t_0}^{t}} \left[R(\nu) + \gamma \sum_{\nu' \in \nu.\text{children}} p(\nu' \mid \nu)\, \text{Minimax}(\nu', D)\right]$ ▷ chance node at time $t$
  **end if**
**end function**

## The Blessing of Optimism

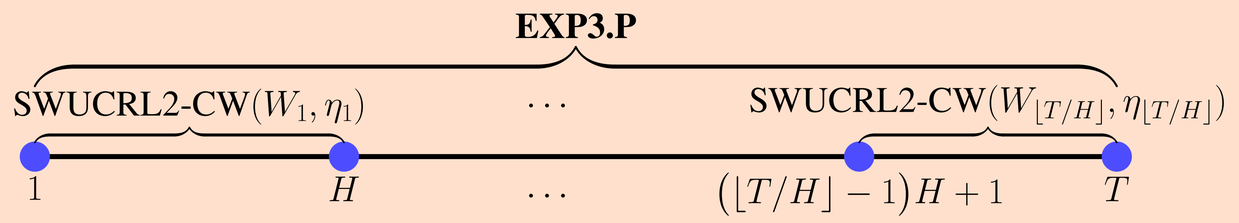
**Problem.** In drifting MDPs, naive sliding-window optimism can pick "optimistic" models with exploding diameter, giving bad dynamic regret. The fix is to add **confidence widening** (extra optimism [2]) to the transition sets.
**Confidence widening.** For windowed estimates $\hat{r}_t, \hat{p}_t$, use

$$\mathcal{H}_{r,t}(s,a) = \{\, \tilde{r} : |\tilde{r} - \hat{r}_t| \leq \text{rad}_{r,t} \,\}, \qquad \mathcal{H}_{p,t}(s,a;\eta) = \{\, \tilde{p} : \|\tilde{p} - \hat{p}_t\|_1 \leq \text{rad}_{p,t} + \eta \,\},$$

with widening $\eta > 0$. This extra optimism is *crucial* under drift.
**Guarantee.** With tuned window $W$ and widening $\eta$,

$$\tilde{\mathcal{O}}\Big(\big(D_{\max}(B_r + B_p)\big)^{1/4} S^{2/3} A^{1/2} T^{3/4}\Big)$$

dynamic regret. BORL (Bandit-over-RL) learns $(W, \eta)$ online and matches the same bound *without* knowing $(B_r, B_p)$.
**Why "more" optimism?** Widening keeps the optimistic model's *effective diameter* controlled ($\approx D_{\max}$), avoiding the drift pitfall of tight sets.




**Algorithm 2** SWUCRL2–CW algorithm

**Require:** Time horizon $T$, state space $\mathcal{S}$, action space $\mathcal{A}$, window size $W$, widening parameter $\eta$
Initialize $t \leftarrow 1$, initial state $s_1$
**for** episode $m = 1, 2, \ldots$ **do**
  Set $\tau(m) \leftarrow t$, $\nu_m(s,a) \leftarrow 0$, and $N_{\tau(m)}(s,a)$ according to Eqn. (3), for all $s, a$
  Compute confidence regions $H_{r,\tau(m)}$, $H_{p,\tau(m)}(\eta)$ according to Eqns. (4,5)
  Compute a $1/\sqrt{\tau(m)}$-optimal optimistic policy

$$\tilde{\pi}_m \leftarrow \text{EVI}(H_{r,\tau(m)}, H_{p,\tau(m)}(\eta); 1/\sqrt{\tau(m)})$$

  **while** $t$ is not a multiple of $W$ and $\nu_m(s_t, \tilde{\pi}_m(s_t)) < N^{+}_{\tau(m)}(s_t, \tilde{\pi}_m(s_t))$ **do**
    Choose action $a_t \leftarrow \tilde{\pi}_m(s_t)$, observe reward $R_t(s_t, a_t)$ and the next state $s_{t+1}$
    Update $\nu_m(s_t, a_t) \leftarrow \nu_m(s_t, a_t) + 1$, $t \leftarrow t + 1$
    **if** $t > T$ **then**
      Terminate algorithm
    **end if**
  **end while**
**end for**

## Results

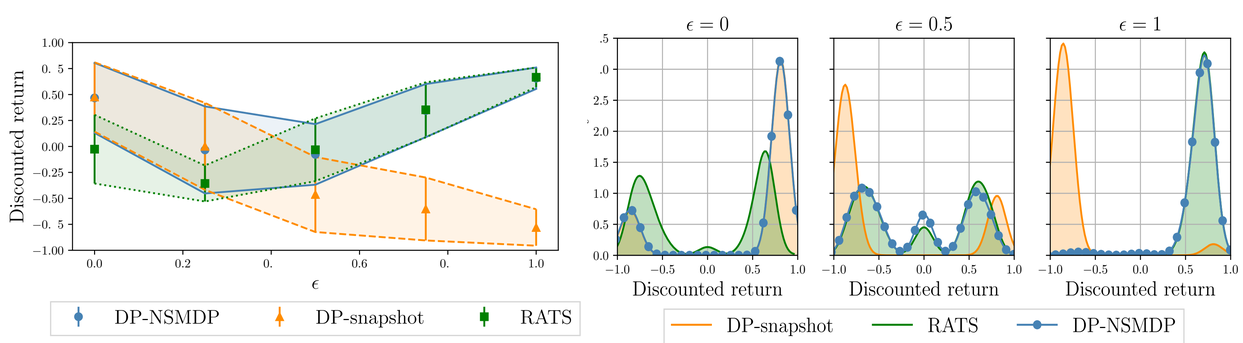



Figure 2: Discounted return of the three algorithms for various values of $\epsilon$.

| $\epsilon$ | | RATS | DP-snapshot | DP-NSMDP |
|---|---|---|---|---|
| 0 | $\mathbb{E}[\sum r]$ | -0.026 | **0.48** | **0.47** |
| | CVaR | **-0.81** | -0.90 | -0.90 |
| 0.5 | $\mathbb{E}[\sum r]$ | **-0.032** | -0.46 | **-0.077** |
| | CVaR | **-0.81** | -0.90 | **-0.81** |
| 1 | $\mathbb{E}[\sum r]$ | **0.67** | -0.78 | **0.66** |
| | CVaR | **0.095** | -0.90 | **-0.033** |

Table 1: Expected return $\mathbb{E}[\sum r]$ and CVaR at $5\%$.

## References

[1] Erwan Lecarpentier and Emmanuel Rachelson. Non-stationary markov decision processes, a worst-case approach using model-based reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[2] Wang Chi Cheung, David Simchi-Levi, and Ruihao Zhu. Reinforcement learning for non-stationary markov decision processes: The blessing of (more) optimism, 2020.