Motivation: RL with limited adaptivity?
- In many domains (recommendation, medical, ...), deploying a new policy is more prohibitive than gathering data with the existing policy.

Online RL is fully adaptive. Any middleground?

Offline (batch) RL is non-adaptive, but much more challenging.

Proposed framework: **low switching cost RL**

**Setup:** Episodic MDP with horizon H. RL algorithm plays K episodes (T = K*H steps.) Measure PAC/Regret.

**Definition:** the switching cost between two (deterministic) policies is number of different actions they would take, (summed) for all (h, s):

\[ n_{\text{switch}}(\pi, \pi') := \sum_{h \in \{1, \ldots, H\}} n_h \neq n_h' \]

**Definition:** the switching cost of an RL algorithm that plays with policies is

\[ N_{\text{switch}} := \sum_{k=1}^{K} n_{\text{switch}}(\pi^k, \pi^{k+1}) \]

Goal: fast exploration with low switching cost

**Prior work:** Q-Learning with UCB exploration:

\[ \hat{Q} \left( \sqrt{\frac{3SA}{K}} \right) \text{regret, but } N_{\text{switch}} = \Theta(HSK) \text{ linear in } K \]  

[1] Jin et al. 2018

Any low regret algorithm such that \( N_{\text{switch}} \) sublinear in K?

Recap: UCB2 scheduling for bandits

**Algorithm** (UCB2): Repeat until played K times:
- Select the arm that maximizes the UCB
- If this is the r-th time it’s selected, play the arm exactly \( \tau(r) = (1 + \alpha)' \) times, where \( \tau(r) = (1 + \alpha)' \)

**Theorem** [Auer et al. 2002]: UCB2 achieves same regret as UCB, and only \( \log(K) \) policy switches:

\[ N_{\text{switch}} = O(A \log(K/A)) \]

Idea: Integrate UCB2 into Q-Learning!

Our Algorithm: **Q-Learning with UCB2 scheduling**

**Key idea:** update the policy only when Q has been updated \( \tau(r) = (1 + \alpha)' \) times.

**Definition:** The triggering sequence \( \{t_n\}_{n \geq 1} \) with parameter \( (\alpha, r) \) is

\[ \{t_n\}_{n \geq 1} = \{1, 2, \ldots, \tau(r)\} \cup \{\tau(r) + 1, \tau(r) + 2, \ldots\} \]

**Algorithm 2** Q-Learning with UCB2-Hoeffding (UCB2H) Exploration.

**Setup**
- Parameter \( n \in \{0, 1\} \), \( r \in [2, \ldots, H] \).
- Initial: \( \hat{Q}_h(x, a) := H, \hat{Q}_h(x, a) \leftarrow 0 \) for all \( x, a, h \) in \( S \times \mathcal{A} \times [H] \).

**For episode k = 1, 2, \ldots, K do**

Receive \( x_h \) for step \( h = 1, \ldots, H \) do
- Take action \( a_h \leftarrow \arg\max_a \hat{Q}_h(x_h, a) \), and observe \( x_{h+1} \). // Take action according to Q
- \( t = \max_{t'=t} x_{h}(t+1) + 1 \). // Update Q via Q-Learning
- \( \hat{V}_h(x_h) \leftarrow \max_a \hat{Q}_h(x_h, a) + 1 + (1 - \alpha)Q_h(x_h, a) + \alpha x_{h+1}(t+1) + 1 \). // Update Q via Q-Learning
- \( \hat{V}_h(x_h) \leftarrow \max_a \hat{Q}_h(x_h, a) \). // Update policy \( Q_{h}(x_h) \leftarrow \hat{V}_h(x_h) \). // Set Q to be Q occasionally according to UCB2 scheduling

**Theoretical Result**

**Theorem 1:** Our Q-Learning with UCB2-[Hoeffding, Bernstein] exploration achieves \( \hat{O} \left( \sqrt{\frac{HS^2A}{K}} \right) \) and logarithmic switching cost:

\[ N_{\text{switch}} \leq O(H^2SA \log(K/A)) \]

Proof highlight: analysis of error propagation under delayed Q updates.

Application: concurrent /parallel RL

**Setup:** M agents play an episode in parallel, and can only communicate after each episode.

**Theorem 2** (Nearly linear speedup in PAC concurrent RL): There exists concurrent versions of our algorithm, s.t. given M agents, it can find \( \varepsilon \) optimal policy in \( \hat{O} \left( \frac{H^2SA}{\varepsilon^2M} \right) \) rounds.

→ Also improves upon prior work [Guo et al. 2015] in (H, \( S, \varepsilon \)) dependence.

**Lower bound on low-switching algs**

Simple Observation: you “need” to switch \( HS(A-1) \) times to at least try out all the possible actions.

**Theorem 3 (Lower bound):** Any algorithm that has switching cost \( N_{\text{switch}} \leq HSA/2 \) has to suffer from linear (trivial) worst-case regret:

\[ \sup_{M \in M} \mathbb{E}[\text{Regret}(K)] \geq KH/4 \]

Remark: Our algorithm achieves \( N_{\text{switch}} = \hat{O}(H^2SA) \), so still an \( H^2 \) gap between the lower and upper bounds.

**Discussion & future work**
- Close the gap on the switching cost.
- Alternative notions of limited adaptivity:
  - Hard constraint on switching cost.
  - RL with only O(1) rounds of adaptivity.
- Connections to fully offline/batch RL.