# The complexity of non-stationary reinforcement learning

**Binghui Peng** BP2601@COLUMBIA.EDU
*Columbia University*

**Christos Papadimitriou** CHRISTOS@COLUMBIA.EDU
*Columbia University*

**Editors:** Claire Vernade and Daniel Hsu

## Abstract

The problem of continual learning in the domain of reinforcement learning, often called non-stationary reinforcement learning, has been identified as an important challenge to the application of reinforcement learning. We prove a worst-case complexity result, which we believe captures this challenge: Modifying the probabilities or the reward of a single state-action pair in a reinforcement learning problem requires an amount of time almost as large as the number of states in order to keep the value function up to date, unless the strong exponential time hypothesis (SETH) is false; SETH is a widely accepted strengthening of the P $\neq$ NP conjecture. Recall that the number of states in current applications of reinforcement learning is typically astronomical. In contrast, we show that just *adding* a new state-action pair is considerably easier to implement.

**Keywords:** Non-stationary reinforcement learning, fine-grained complexity

## 1. Introduction

Reinforcement learning (RL) (Sutton and Barto, 2018), the branch of machine learning seeking to create machines that react to a changing environment so as to maximize long-term utility, has recently seen tremendous advances through deep learning (Silver et al., 2017, 2018), as well as a vast expansion of its applicability and reach to many application domains, including board games, robotics, self-driving cars, control, and many more. As with most aspects of deep learning, one of the most important current challenges in deep RL lies in handling situations in which the model undergoes changes. Variably called *non-stationary RL, continual RL, multi-task RL, or life-long RL*, the problem of enabling RL to react effectively and gracefully to sequences of changes in the underlying Markov model has been identified as an important open problem in practice, see the prior work subsection for many references, and Khetarpal et al. (2022) for a recent survey of the challenge and the available remedies.

When it becomes clear that a particular computational problem is difficult, the field of *computational complexity* (Papadimitriou and Steiglitz, 1998; Papadimitriou, 2003; Arora and Barak, 2009) comes into play: the search for mathematical obstacles to the efficient solution of problems. The identification of such obstacles is often informative about the kinds of remedies one needs to apply to the problem. As far as we can tell, the computational complexity of non-stationary RL (NSRL) has not been explored in the past; in contrast, see Chen et al. (2022) for an example of recent progress in identifying complexity obstacles in continual learning of *classification* tasks.

*In this paper, we initiate the analysis of NSRL from the standpoint of computational complexity.* We consider finite horizon MDPs — it is easy to see that our results can be extended very easily to infinite horizon MDPs. We ask the following question: Suppose that we have already solved a finite-horizon MDP, and that the MDP changes in some small way; how difficult is it to modify the

solution? If the solution we want to update is an explicit mapping from states to actions, then it is not hard to see that this problem is hopeless: a small local change can cause a large proportion of the values of this map to change[1]. In practice, however, deep RL is not about computing explicitly the optimum solution of the problem; it is about maintaining an implicit, deep-net representation of a good *approximation* of the optimum solution, and only the parameters of this representation are updated. Our results address precisely this aspect of the difficulty.

We consider elementary local changes to the RL problem, which we believe capture well the nature of the NSRL problem: We choose a state-action pair and we modify somehow its parameters: the reward, and the transition probability distribution. Our results hold for the most elementary possible change: We only modify two transition probabilities in one state-action pair. (Naturally, it is impossible to modify only one probability in a distribution...) We prove that, under widely accepted complexity assumptions to be explained soon, the amount of computation needed to update an $\epsilon$-optimal value approximation in the face of such an elementary change is, in the worst case, comparable to the number of states (the precise result is stated below). Since in the problems currently solved by deep RL the number of states of the underlying MDP is typically astronomical, such a prediction is bad indeed — it means that we essentially have to start all over because of a small change. Now, in deep learning we know well that a worst-case result is never the last word on the difficulty of a problem. However, we believe that an alarming worst-case result, established for an aspect of the problem which has been identified in practice to be a challenge, is a warning sign which may yield valuable hints about the corrective action that needs to be taken in order to overcome the current bottleneck.

We complement this lower bound with a positive result for a different kind of change: *adding a new* action to a state. It turns out that this is a simpler problem, and an $\epsilon$-approximate solution can be updated in time polynomial in $\frac{1}{\epsilon}$ and the horizon.

## Related work

Non-stationary MDPs have been studied extensively in recent years from the point of view of dynamic regret (Auer et al., 2008; Dick et al., 2014; Ortner et al., 2020; Cheung et al., 2020; Zhou et al., 2022; Li and Li, 2019; Touati and Vincent, 2020; Wei and Luo, 2021; Domingues et al., 2021; Mao et al., 2021; Jiang et al., 2023; Feng et al., 2023); In Mao et al. (2021) an algorithm with total regret $\widetilde{O}(S^{1/3}A^{1/3}\Delta^{1/3}HT^{2/3})$ is provided, where $T$ is the total number of iteration, $\Delta$ is the variational budget that measures the total change of MDP. Another line of work (Da Silva et al., 2006; Banerjee et al., 2017; Padakandla et al., 2020; Ornik and Topcu, 2021) focuses on the statistical problem of detecting the changes in the environment. We refer interested reader to Padakandla (2021); Khetarpal et al. (2022) for recent surveys; in particular, Padakandla (2021) mentions the computational difficulty of the change problem, which is an important yet unresolved open question in the literature. Several approaches to NSRL — e.g Wei and Luo (2021); Mao et al. (2021) — resort to *restarting* the learning process if enough change has accumulated; our results suggest that, indeed, restarting may be preferable to updating. Additional literature can be found at Appendix A.

1. For example, consider the extreme example where a change in an action increases the value of the next state, and this in turn changes the optimum actions in almost all other states.

### A brief overview of the main result

In the rest of the paper we use the $o(1)$ notation; in particular, $\Omega(n^{1-o(1)})$ is the class of functions that is at least $n^{1-\epsilon}$ for any $\epsilon > 0$ when $n \to +\infty$, and $n^{o(1)}$ is the class of functions that is no more than $n^{\epsilon}$ for any $\epsilon > 0$ when $n \to +\infty$

Our main result (Theorem 1) states that, in the worst case, an elementary change in an MDP — just updating two transition probabilities in one action at one state of the MDP — requires time $\Omega((SAH)^{1-o(1)})$, where $S$ is the number of states, $A$ is the number of actions and $H$ is the horizon. The proof is based on the Strong Exponential Time Hypothesis (SETH), which is a central conjecture in complexity, a refinement of P $\neq$ NP. SETH has many applications in graph algorithms (Roditty and Vassilevska Williams, 2013; Abboud and Williams, 2014; Backurs et al., 2018; Li, 2021; Dalirrooyfard et al., 2022), edit distance (Backurs and Indyk, 2015), nearest neighbor search (Rubinstein, 2018), kernel estimation (Charikar and Siminelakis, 2017; Alman et al., 2020) and many other domains; see (Rubinstein and Williams, 2019) for a comprehensive survey. SETH states that, if the $k$-SAT problem (the Boolean satisfiability problem when each clause contains at most $k$ literals) can be solved in time $O(2^{c_k n})$, then the limit of $c_k$ as $k$ grows is one. Our work is based on the important result of (Abboud et al., 2017) on the hardness, under SETH, of approximating the bichromatic Maximum Inner Product (MAX-IP) problem. Subsequent work has improved the approximation parameter (Rubinstein, 2018; Chen, 2020) and applied the technique to the Dynamic Coverage problem (Abboud et al., 2019; Peng, 2021).

We reduce from the MAX-IP problem, where we are given two collections of sets $B_1, \ldots, B_n$ and $C_1, \ldots, C_n$, over a small universe $[m]$ with $m = n^{o(1)}$. It is known from Abboud et al. (2017) that it is hard to distinguish between the following two scenaria: (a) $B_i \subseteq C_j$ for some $i, j \in [n]$, and (b) $|B_i \cap C_j| \leq |C_j|/2^{\log(n)^{1-o(1)}}$ for all $i, j \in [n]$. That is, it is hard to tell the difference between the case of a complete containment and the case of tiny intersections. The first step of our reduction is to construct a finite-horizon MDP such that the state of the first step ($h = 1$) corresponds to the sets $B_1, \ldots, B_n$ and the state of the second step ($h = 2$) corresponds to the universe $[m]$. The state of the second step has either high reward or low reward, depending on the time $t$. By applying a sequence of changes to the state-action transition in the second step, based on the structure of the sets $C_1, \ldots, C_n$, one obtains a reduction from MAX-IP establishing a lower bound of $\Omega(S^{2-o(1)})$ for this sequence. However, since this sequence is of length $S^{1+o(1)}$ (because of the size of the $C_j$ sets), we obtain an $\Omega(S^{1-o(1)})$ amortized lower bound for each step of the sequence, and this completes the reduction to the NSRL problem.

The construction so far yields an approximation $\epsilon$ that is very small (i.e., $S^{-0.001}$). We need a second stage of our construction to amplify $\epsilon$ to some constant such as 0.1. This is achieved by stacking multiple layers of the basic construction outlined above. Finally, by spreading the state-actions across multiple steps, we improve the lower bound to $\Omega((SAH)^{1-o(1)})$.

The complete proof can be found at Section 3.

## 2. Preliminary Definitions

Here we shall define non-stationary MDPs. Let $\mathcal{S}$ be a state space ($|\mathcal{S}| = S$), $\mathcal{A}$ an action space ($|\mathcal{A}| = A$), $H \in \mathbb{Z}_+$ the planning horizon. Next let $T \in \mathbb{Z}_+$ be the number of *rounds:* The intention is that the MDP will repeat $T$ times, with action parameters changed between rounds.

A *non-stationary finite horizon MDP* is a set of $T$ MDPs $(\{\mathcal{S}_h, \mathcal{A}_h, P_{t,h}, r_{t,h}\}_{t\in[T],h\in[H]}, s_{\text{init}})$. $\mathcal{S}_h \subseteq \mathcal{S}$ is the state space and $\mathcal{A}_h \subseteq \mathcal{A}$ is the action space at the $h$-th step ($h \in [H]$), and $P_{t,h}$ :

$\mathcal{S}_h \times \mathcal{A}_h \to \Delta_{\mathcal{S}_{h+1}}$ is the transition function, where $\Delta_{\mathcal{S}_h}$ is the set of all probability distributions over $\mathcal{S}_h$, and $r_{t,h} : \mathcal{S}_h \times \mathcal{A}_h \to [0,1]$ is the reward function at the $h$-th step of the $t$-th round ($h \in [H], t \in [T]$). We use $s_{\text{init}} \in \mathcal{S}_1$ to denote the initial state.

We focus on deterministic non-stationary policies $\pi = (\pi_1, \ldots, \pi_T)$, though our results can be applied to randomized policies as well. Let $\pi_t = (\pi_{t,1}, \ldots, \pi_{t,H})$ be the policy of the $t$-th round ($t \in [T]$) and $\pi_{t,h} : \mathcal{S}_h \to \mathcal{A}_h$ ($h \in [H]$) be the decision at the $h$-th step. Given a policy $\pi$, the $Q$-value of a state-action pair $(s,a) \in \mathcal{S}_h \times \mathcal{A}_h$ at the $t$-round can be determined

$$Q_{t,h}^{\pi_t}(s,a) = r_{t,h}(s,a) + \mathbb{E}\left[\sum_{\ell=h+1}^{H} r_{t,h}(s_{t,\ell}, \pi_{t,\ell}(s_{t,\ell})) \mid s_{t,h} = s, a_{t,h} = a\right] \quad \forall s \in \mathcal{S}_h, a \in \mathcal{A}_h$$

and the $V$-value

$$V_{t,h}^{\pi_t}(s) = \mathbb{E}\left[\sum_{\ell=h}^{H} r_{t,h}(s_{t,\ell}, \pi_{t,\ell}(s_{t,\ell})) \mid s_{t,h} = s\right] \quad \forall s \in \mathcal{S}_{t,h}.$$

Let $\pi_t^*$ be the optimal policy at the $t$-round, and $Q_t^*$, $V_t^*$ be its $Q$-value and $V$-value. The goal is to maintain an $\epsilon$-approximate value function. In particular, we require that an approximation $V_t$ of the value of the initial state $s_{\text{init}}$ be maintained, such that for all rounds $t \in [T]$,

$$\left|V_t - V_{t,1}^*(s_{\text{init}})\right| \le \epsilon.$$

**Updates.** All $T$ MDPs of our definition must be solved, one after the other, in the face of parameter changes that are meant to be extremely simple and local: For the $t$-th update, an adversary picks an arbitrary state-action pair $(s_h, a_h) \in \mathcal{S}_h \times \mathcal{A}_h$, and changes the transition function from $P_{t-1,h}(s_h, a_h)$ to $P_{t,h}(s_h, a_h)$ and the reward from $r_{t-1,h}(s_h, a_h)$ to $r_{t,h}(s_h, a_h)$. It also changes the transition function from $P_{t-1,h}(s_h, a_h)$ to $P_{t,h}(s_h, a_h)$, such that these two distributions differ in *exactly two states.* That is, the change in the distribution is the smallest kind imaginable: *Two next states are chosen, and the probability mass of the first is transferred to the second.* Since the changes we consider are the simplest possible, our proof that even these changes are computationally intractable leaves little hope for the worst-case of NSRL.

## 3. Hardness of NSRL

The main result is the following:

**Theorem 1 (Main result, hardness of NRSL)** *Let $S, A, H$ be sufficiently large integers, and $T$ is bounded by arbitrarily large polynomials $T \le \text{poly}(SAH)$, while the horizon $H \ge (SA)^{o(1)}$. Then, unless* SETH *is false, there is no algorithm with amortized runtime $O((SAH)^{1-o(1)})$ per update that can approximate the optimal value of a non-stationary MDP over a sequence of $T$ updates. In particular, any algorithm with better runtime fails to distinguish between these two cases:*

- *The optimal policy has value at least $\frac{H}{4}$ at some round $t \in [T]$;*
- *The optimal policy has value at most $\frac{H}{100}$ for all $T$ rounds.*

**Remark 2 (Sequence length and horizon)** *In the statement of Theorem 1 we assume that $T$ is polynomially bounded. Since $T$ is the sequence of MDPs presented for solution, it would be unreasonable to be exponential, especially in $S$, which is typically huge. Furthermore, proving lower bounds in the face of exponential $T$ would be impossible, for example, through the slow design of a look-up table that solves the problem. The assumption $H \geq (SA)^{o(1)}$ is not needed, and our result holds even if $H$ is a constant, but the gap would be smaller than $\frac{H}{4}$ vs. $\frac{H}{100}$.*

Our result is based on the widely accepted Strong Exponential Time Hypothesis (SETH).

**Conjecture 3 (Strong Exponential Time Hypothesis (SETH), Impagliazzo and Paturi (2001))** *For any $\delta > 0$, there exists $k \geq 3$ such that the $k$-SAT problem on $n$ variables cannot be solved in time $O(2^{(1-\delta)n})$.*

**Remark 4 (Why we need SETH)** *The use of a hypothesis stronger than "$\mathsf{P} \neq \mathsf{NP}$" is needed, because it is known that "$\mathsf{P} \neq \mathsf{NP}$" cannot yield any complexity lower bounds within $\mathsf{P}$ (where NSRL belongs), see the survey of Williams (2018).*

The starting point of our reduction is the following hardness result for the Bichromatic Maximum Inner Product (MAX-IP) problem, whose proof is based on the machinery of distributed PCP.

**Theorem 5 (Bichromatic Maximum Inner Product (MAX-IP) Abboud et al. (2017))** *Let $\gamma > 0$ be any constant, and let $n \in \mathbb{Z}_+$, $m = n^{o(1)}$, $w = 2^{(\log(n))^{1-o(1)}}$. Given two collections of sets $\mathcal{B} = \{B_1, \ldots, B_n\}$ and $\mathcal{C} = \{C_1, \ldots, C_n\}$ over universe $[m]$, satisfying $|B_1| = \cdots = |B_n| = b$ and $|C_1| = \cdots = |C_n| = c$ for some $b, c \in [m]$. Unless SETH is false, no algorithm can distinguish the following two cases in time $O(n^{2-\gamma})$:*

- **YES instance.** *There exists two sets $B \in \mathcal{B}$, $C \in \mathcal{C}$ such that $C \subseteq B$;*
- **NO instance.** *For every $B \in \mathcal{B}$ and $C \in \mathcal{C}$, $|B \cap C| \leq c/w$.*

**Parameters.** We reduce MAX-IP to NSRL. For any sufficiently large parameters $S, A, H, T$, let

$$n = T^{1/2-o(1)} \cdot (SAH)^{1/2} \quad \text{and} \quad m = n^{o(1)}$$

be the input parameters of MAX-IP. Given a MAX-IP instance with sets $B_1, \ldots, B_n$ and $C_1, \ldots, C_n$ over a ground set $[m]$, recall $b, c \in [m]$ are the size of set $\{B_i\}_{i\in[n]}$ and $\{C_i\}_{i\in[n]}$ . Let

$$L = \lceil b/c \rceil \quad \text{and} \quad N = \frac{SAH}{16L(\log_2(S) + 2)}.$$

We shall divide $\{B_i\}_{i\in[n]}$ into $K = n/N$ batches and each batch contains $N$ sets. That is, $\{B_i\}_{i\in[n]} = \{B_{k,\nu}\}_{k\in[K],\nu\in[N]}$.

### 3.1. Construction of a hard instance

We first describe the MDP at the initial stage ($t = 0$), with state space $\{\mathcal{S}_h\}_{h\in[H]}$, action space $\{\mathcal{A}_h\}_{h\in[H]}$, transition function $\{P_h\}_{h\in[H]}$ and reward function $\{r_h\}_{h\in[H]}$. A (simplified) illustration can be found at Figure 1. We omit the subscript of $t = 0$ for simplicity.

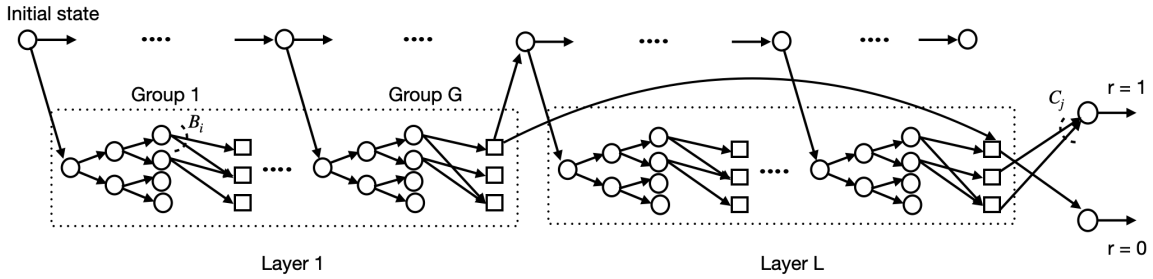


Figure 1: A snapshot of the hard instance

**Horizon.** We divide the entire horizon into two phases

$$[H] = \mathcal{H}_1 \cup \mathcal{H}_2, \quad \text{where} \quad \mathcal{H}_1 = [H/2] \quad \text{and} \quad \mathcal{H}_2 = [H/2 : H]\,.$$

The second phase is relatively simple and involves only two terminal states that provide rewards. The first phase is more involved and determines the destination state.

The first phase contains $L$ layers, and each layer contains $H/2L$ steps

$$\mathcal{H}_1 = \mathcal{H}_{1,1} \cup \cdots \cup \mathcal{H}_{1,L}, \quad \text{where} \quad \mathcal{H}_{1,\ell} = \left[(\ell-1)\cdot\frac{H}{2L} + 1 : \ell \cdot \frac{H}{2L}\right] \ \forall \ell \in [L].$$

The layers are used for amplifying the difference between good and bad policies. The structure of the MDP is for identical for each layer, except the last step at the last layer.

For each layer $\ell \in [L]$, we further divide it into $G := \frac{H}{2L(\log_2(S)+2)}$ groups, and each group contains $\log_2(S)+2$ steps,

$$\mathcal{H}_{1,\ell} = \mathcal{H}_{1,\ell,1} \cup \cdots \cup \mathcal{H}_{1,\ell,G}$$

where

$$\mathcal{H}_{1,\ell,g} = \left[(\ell-1)\cdot\frac{H}{2L} + (g-1)(\log_2(S)+2) + 1 : (\ell-1)\cdot\frac{H}{2L} + g\cdot(\log_2(S)+2)\right] \ \forall g \in [G].$$

We set $h(\ell, g, \tau) := (\ell-1)(H/2L) + (g-1)(\log_2(S)+2) + \tau$ be the $\tau$-step, at the $g$-th group of the $\ell$-th layer, where $\tau \in [\log_2(S)+2], g \in [G], \ell \in [L]$. For simplicity, we also write $h(\ell, g) = h(\ell, g, \log_2(S)+2)$ and $h(\ell) = h(\ell, G)$ be the last step of each group and layer.

**States.** There are five types of states: terminal states, element states, set states, routing states and the pivotal state.

- **Terminal states.** There are two terminal states $s_1^{\mathsf{t}}$ and $s_2^{\mathsf{t}}$, and they appear at every steps $h \in [H]$. We use $s_{h,1}^{\mathsf{t}}, s_{h,2}^{\mathsf{t}}$ to denote the terminal states at $\mathcal{S}_h$.

- **Element states.** There are $m$ element states $\{s_u^{\mathsf{e}}\}_{u\in[m]}$ that appear at every step $h \in \mathcal{H}_1$ of phase one. We use $s_{h,u}^{\mathsf{e}}$ to denote the $u$-th element state at $\mathcal{S}_h$.

- **Set states.** There are $S/4$ set states $\{s_i^{\mathsf{b}}\}_{i\in[S/4]}$. The set states only appear on the second last step of each group $\mathcal{H}_{\ell,g}$. In particular, for each layer $\ell \in [L]$, group $g \in [G]$, let $s_{h(\ell,g)-1,i}^{\mathsf{b}}$ denote the $i$-th ($i \in [S/4]$) set state at $\mathcal{S}_{h(\ell,g)-1}$.
- **Pivotal state** There is one pivotal state $s^{\mathsf{p}}$ that appears at every step $h \in \mathcal{H}_1$ of Phase 1, denoted as $s_h^{\mathsf{p}}$. The MDP start with the pivotal state, i.e., $s_{\text{init}} := s_1^{\mathsf{p}}$.
- **Routing states.** The routing states are used for reaching set states. There $S/4$ routing states $\{s_\alpha^{\mathsf{r}}\}_{\alpha\in[S/4]}$ that appear at the $[2 : \log_2(S)]$-th step of each group. In particular, at layer $\ell \in [L]$, group $g \in [G]$, step $\tau \in [2 : \log_2(S)]$, let $\{s_{h(\ell,g,\tau),\alpha}^{\mathsf{r}}\}_{\alpha\in[1:2^{\tau-2}]}$ be the collection of routing states at $\mathcal{S}_{h(\ell,g,\tau)}$.

The total number of possible states is at most $2 + m + S/4 + S/4 + 1 \le S$.

**Actions** There are five types of actions. The terminal action $a^{\mathsf{t}}$, the element actions $a^{\mathsf{e}}$, the set actions $\{a_j^{\mathsf{b}}\}_{j\in[A/2]}$, the pivotal action $\{a_1^{\mathsf{p}}, a_2^{\mathsf{p}}\}$ and the routing actions $\{a_1^{\mathsf{r}}, a_2^{\mathsf{r}}\}$. The total number of action is at most $A/2 + 6 \le A$, and we assume these actions appear at every step $h \in [H]$.

**Reward** The only state that returns non-zero reward is the terminal state $\{s_{h,1}^{\mathsf{t}}\}_{h\in\mathcal{H}_2}$. Formally, we set

$$r_h(s,a) = 0 \quad \text{when} \quad h \in \mathcal{H}_1 \quad \text{and} \quad r_h(s,a) = \begin{cases} 1 & s = s_{h,1}^{\mathsf{t}} \\ 0 & \text{otherwise} \end{cases} \quad \text{when} \quad h \in \mathcal{H}_2. \tag{1}$$

**Transitions** We next specify the transition probability of the initial MDP.

**(a) Terminal states.** The transition of terminal states is deterministic and always keeps the state terminal, that is

$$P_h(s_{h,1}^{\mathsf{t}}, a) = \mathbf{1}\{s_{h+1,1}^{\mathsf{t}}\} \quad \text{and} \quad P_h(s_{h,2}^{\mathsf{t}}, a) = \mathbf{1}\{s_{h+1,2}^{\mathsf{t}}\} \quad \forall h \in [H-1], a \in \mathcal{A}. \tag{2}$$

Here we use $\mathbf{1}\{s\} \in \Delta_{\mathcal{S}_{h+1}}$ to denote the one-hot vector that is 1 at state $s$ and 0 otherwise. Combining with the definition of reward functions, the MDP guarantees that a policy receives $H/2$ reward once it goes to the first terminal state $s_{h,1}^{\mathsf{t}}$ at some step $h \in \mathcal{H}_2$. Meanwhile, it receives 0 reward if it ever goes to the second terminal state $s_{h,2}^{\mathsf{t}}$.

**(b) Element states.** At step $h < H/2$, for any element $u \in [m]$, the transition function of $s_{h,u}^{\mathsf{e}}$ equals

$$P_h(s_{h,u}^{\mathsf{e}}, a^{\mathsf{e}}) = \begin{cases} \mathbf{1}\{s_{h+1}^{\mathsf{p}}\} & h = h(\ell) \text{ for some } \ell \in [L-1] \\ \mathbf{1}\{s_{h+1,u}^{\mathsf{e}}\} & \text{otherwise} \end{cases} \tag{3}$$

and

$$P_h(s_{h,u}^{\mathsf{e}}, a) = \mathbf{1}\{s_{h+1,u}^{\mathsf{e}}\}, \quad \forall a \in \mathcal{A}\backslash\{a^{\mathsf{e}}\}. \tag{4}$$

That is, the element state $s_{h,u}^{\mathsf{e}}$ always stays on itself, except at the end of each layer $\ell \in [L]$, it can go to the pivotal state.

At the end of the first phase, the transition of element state is determined by the set $C$. In the initialization stage ($t = 0$), let $C_0 \subseteq [m]$ be an arbitrary set of size $c$ and it would be replace later, let

$$P_{H/2}(s_{H/2,u}^{\mathsf{e}}, a^{\mathsf{e}}) = \begin{cases} \mathbf{1}\{s_{H/2+1,1}^{\mathsf{t}}\} & u \in C_0 \\ \mathbf{1}\{s_{H/2+1,2}^{\mathsf{t}}\} & u \notin C_0 \end{cases}$$

and

$$P_{H/2}(s^{\mathsf{e}}_{H/2,u}, a) = \mathbf{1}\{s^{\mathsf{t}}_{H/2+1,2}\}, \quad \forall a \in \mathcal{A}\backslash\{a^{\mathsf{e}}\}.$$

That is, if the element $u \in C_0$, then it can go to a high reward terminal state $s^{\mathsf{t}}_{H/2+1,1}$; otherwise it goes to the no-reward terminal $s^{\mathsf{t}}_{H/2+1,2}$. Looking ahead, we would update the state-action pairs $\{(s^{\mathsf{e}}_{H/2,u}, a^{\mathsf{e}})\}_{u\in[m]}$ according to sets $\{C_i\}_{i\in[n]}$ periodically.

**(c) Set states.** The transition function of set states is determined by the sets $\{B_{k,\nu}\}_{k\in[K],\nu\in[N]}$. In the initialization stage ($t = 0$), let $\{B_{0,\nu}\}_{\nu\in[N]}$ be arbitrary sets of size $b$ and they would be replaced later in the update sequence. Recall that a set state would appear at the second last step of a group $\mathcal{H}_{\ell,g}$, for some layer $\ell \in [L]$ and group $g \in [G]$. Let

$$N(g, i, j) := (g - 1)(S/4)(A/2) + (i - 1)(A/2) + j,$$

and therefore,

$$\{N(g, i, j) : g \in [G], i \in [S/4], j \in [A/2]\} = [N].$$

The transition function of state-action pair $(s^{\mathsf{b}}_{h(\ell,g)-1,i}, a^{\mathsf{b}}_j)$ equals

$$P_{h(\ell,g)-1}(s^{\mathsf{b}}_{h(\ell,g)-1,i}, a^{\mathsf{b}}_j) = \mathrm{unif}(s^{\mathsf{e}}_{h(\ell,g),u} : u \in B_{0,N(g,i,j)}) \quad \forall g \in [G], i \in [S/4], j \in [A/2]. \tag{5}$$

Here the RHS is the uniform distribution over the element states $s^{\mathsf{e}}_{h(\ell,g),u}$ for element $u \in B_{0,N(g,i,j)}$. For the rest of actions, it goes to the no-reward terminal $s^{\mathsf{t}}_{h(\ell,g),2}$:

$$P_{h(\ell,g)-1}(s^{\mathsf{b}}_{h(\ell,g)-1,i}, a) = \mathbf{1}\{s^{\mathsf{t}}_{h(\ell,g),2}\} \quad \forall a \in \mathcal{A}\backslash\{a^{\mathsf{e}}_j\}_{j\in[A/2]}$$

**(d) Pivotal states.** The pivotal state $s^{\mathsf{p}}_h$ appears at every step $h \in \mathcal{H}_1$, and for $h < H/2 - 1$, the transition function equals

$$P_h(s^{\mathsf{p}}_h, a) = \begin{cases} \mathbf{1}\{s^{\mathsf{r}}_{h+1,1}\} & a = a^{\mathsf{p}}, h = h(\ell, g, 1) \text{ for some } \ell \in [L], g \in [G] \\ \mathbf{1}\{s^{\mathsf{p}}_{h+1}\} & \text{otherwise} \end{cases} \tag{6}$$

That is, the pivotal state stays on itself, except at the first step of $\mathcal{H}_{\ell,g}$, it could go to the routing state $s^{\mathsf{r}}_{h(\ell,g,2),1}$.

At the $H/2$-th step, it goes to the no-reward terminal $s^{\mathsf{t}}_{H/2+1,2}$,

$$P_{H/2}(s^{\mathsf{p}}_{H/2}, a) = \mathbf{1}\{s^{\mathsf{t}}_{H/2+1,2}\} \quad \forall a \in A.$$

**(e) Routing states.** Recall $\{s^{\mathsf{r}}_{h(\ell,g,\tau),\alpha}\}_{\alpha\in[1:2^{\tau-2}]}$ is the collection of routing states at the $\alpha$-th step ($\alpha \in [2 : \log_2(S)]$), $g$-th group ($g \in [G]$) and $\ell$-th layer ($\ell \in [L]$).

When $\tau \in [2 : \log_2(S) - 1]$, the transition function equals

$$P_{h(\ell,g,\tau)}(s^{\mathsf{r}}_{h(\ell,g,\tau),\alpha}, a) = \begin{cases} 1\{s^{\mathsf{r}}_{h(\ell,g,\tau+1),2\alpha-1}\} & a = a^{\mathsf{r}}_1 \\ 1\{s^{\mathsf{r}}_{h(\ell,g,\tau+1),2\alpha}\} & a = a^{\mathsf{r}}_2 \\ 1\{s^{\mathsf{t}}_{h(\ell,g,\tau+1),2}\} & \text{otherwise} \end{cases}, \quad \forall \alpha \in [2^{\tau-2}]. \tag{7}$$

In other words, the routing state $s^{\mathsf{r}}_{h(\ell,g,\tau),\alpha}$ goes to either $s^{\mathsf{r}}_{h(\ell,g,\tau+1),2\alpha-1}$ or $s^{\mathsf{r}}_{h(\ell,g,\tau+1),2\alpha-1}$, depending on the choice of actions (unless it goes to the no-reward terminal $s^{\mathsf{r}}_{h(\ell,g,\tau+1),2}$).

When $\tau = \log_2(S)$, the routing state $s^{\mathsf{r}}_{h(\ell,g,\log_2(S)),\alpha}$ goes to the set state $s^{\mathsf{b}}_{h(\ell,g)-1,\alpha}$ ($\alpha \in [S/4]$), that is,

$$P_{h(\ell,g,\log_2(S))}(s^{\mathsf{r}}_{h(\ell,g,\log_2(S)),\alpha}, a) = s^{\mathsf{b}}_{h(\ell,g)-1,\alpha}, \quad \forall \alpha \in [S/4], a \in \mathcal{A}. \tag{8}$$

The entire transition of routing states within a group works like a binary search tree: it comes from the pivotal state and goes to one of the set states. We note that if $S \leq A$ the construction could be simplified: we can remove routing states and have a pivotal state directly go to set states. This completes the description of the initial MDP.

**Update sequence.** We next specify the sequence of updates to the MDP. The sequence of updates is divided into $K = n/N$ stages, and each stage contains $n$-epochs.

At the beginning of each stage, the update occurs on the state-action pairs for set-states:

$$\{(s^{\mathsf{b}}_{h(\ell,g)-1,i}, a^{\mathsf{b}}_j)\}_{\ell\in[L],g\in[G],i\in[S/4],j\in[A/2]}$$

Concretely, there is an initialization phase at the beginning of the $k$-th stage ($k \in [K]$). Let $t(k) \in [T]$ be the end of initiazation phase, and the nature sets

$$P_{t(k),h(\ell,g)-1}(s^{\mathsf{b}}_{h(\ell,g)-1,i}, a^{\mathsf{b}}_j) = \mathrm{unif}(s^{\mathsf{e}}_{h(\ell,g),u} : u \in B_{k,N(g,i,j)}) \quad \forall \ell \in [L], g \in [G], i \in [S/4], j \in [A/2].$$

Each stage contains $n$-epochs, and during each epoch, the update occurs on the state-action pairs $\{(s^{\mathsf{e}}_{H/2,u}, a^{\mathsf{e}})\}_{u\in[m]}$ of element state-action, in the $H/2$-th step. Let $t(k,\tau) \in [T]$ be the end of $k$-th ($k \in [K]$) stage and $\tau$-th ($\tau \in [n]$) epoch. In the $\tau$-th epoch ($\tau \in [n]$), for each element $u \in [m]$, the transition function is updated to

$$P_{t(k,\tau),H/2}(s^{\mathsf{e}}_{H/2,u}, a^{\mathsf{e}}) = \begin{cases} \mathbf{1}\{s^{\mathsf{t}}_{H/2+1,1}\} & u \in C_\tau \\ \mathbf{1}\{s^{\mathsf{t}}_{H/2+1,2}\} & u \notin C_\tau \end{cases}. \tag{9}$$

To count the total number of updates, there are $K = n/N$ stages. The initialization takes at most $O(SAHm)$ updates; there are $n$ epochs, and each epoch contains at most $2m$ updates. Hence the total number of updates equals $(n/N) \cdot O(SAHm + 2mn) \approx T$.

### 3.2. Analysis

We now proceed to prove Theorem 1. For any stage $k \in [K]$ and epoch $\tau \in [n]$, we compute the $V$-value of the optimal policy. The proof can be found at the Appendix B

**Lemma 6 ($V$-value, terminal states)** *At the end of stage $k \in [K]$ and epoch $t \in [n]$, for any step $h \in [H]$, the $V$-value of optimal policy at terminal states satisfies $V^*_{t(k,\tau),h}(s^{\mathsf{t}}_{h,1}) = \min\{H + 1 - h, H/2\}$ and $V^*_{t(k,\tau),h}(s^{\mathsf{t}}_{h,2}) = 0$.*

**Lemma 7 ($V$-value, element states)** *At the end of stage $k \in [K]$ and epoch $\tau \in [n]$, for any layer $\ell \in [L]$ and any step $h \in \mathcal{H}_{1,\ell}$*

- *For any element $u \in C_\tau$, $V^*_{t(k,\tau),h}(s^{\mathsf{e}}_{h,u}) = H/2$; and*
- *For any element $u \notin C_\tau$, we have $V^*_{t(k,\tau),h}(s^{\mathsf{e}}_{h,u}) = V^*_{t(k,\tau),h(\ell)+1}(s^{\mathsf{p}}_{h(\ell)+1})$.*

*Here we take* $V_{t(k,\tau),H/2+1}(s^{\mathsf{p}}_{H/2+1}) := 0$.

**Lemma 8 ($V$-value, set states)** *At the end of stage $k \in [K]$ and epoch $\tau \in [n]$, for each level $\ell \in [L]$, group $g \in [G]$, we have*

$$\begin{aligned} &V^*_{t(k,\tau),h(\ell,g)-1}(s^{\mathsf{b}}_{h(\ell,g)-1,i}) \\ &= \max_{j\in[A/2]} \left\{ \frac{|C_\tau \cap B_{k,N(g,i,j)}|}{b} \cdot \frac{H}{2} + \left(1 - \frac{|C_\tau \cap B_{k,N(g,i,j)}|}{b}\right) \cdot V^*_{t(k,\tau),h(\ell)+1}(s^{\mathsf{p}}_{h(\ell)+1}) \right\} \end{aligned}$$

**Lemma 9 ($V$-value, pivotal state)** *At the end of stage $k \in [K]$ and epoch $\tau \in [n]$, for each level $\ell \in [L]$, the $V$-value of the pivotal state satisfies*

$$\begin{aligned} &V^*_{t(k,\tau),h(\ell-1)+1}(s^{\mathsf{p}}_{h(\ell-1)+1}) \\ &= \max_{\nu\in[N]} \left\{ \frac{|C_\tau \cap B_{k,\nu}|}{b} \cdot \frac{H}{2} + \left(1 - \frac{|C_\tau \cap B_{k,\nu}|}{b}\right) \cdot V^*_{t(k,\tau),h(\ell)+1}(s^{\mathsf{p}}_{h(\ell)+1}) \right\}. \end{aligned}$$

As a corollary, we can compute the $V$-value of the initial state.

**Lemma 10 ($V$-value, initial state)** *Let $\kappa_{k,\tau} = \max_{\nu\in[N]} \frac{|C_\tau \cap B_{k,\nu}|}{b}$, then at the end of stage $k$ and epoch $\tau \in [n]$, one has*

$$V^*_{t(k,\tau),1}(s_{\text{init}}) = (1 - (1 - \kappa_{k,\tau})^L) \cdot \frac{H}{2}.$$

Now we can complete the proof of Theorem 1

**Proof** [Proof of Theorem 1] If the input of MAX-IP is a YES instance, suppose $C_\tau \subseteq B_{k,\nu}$ for some $\tau \in [n], k \in [K], \nu \in [N]$; then $\kappa_{k,\tau} = c/b = 1/L$. By Lemma 10, the value of $s_{\text{init}}$ at the end of epoch $t$ satisfies

$$V^*_{t(k,\tau),1}(s_{\text{init}}) = (1 - (1 - \kappa_{k,\tau})^L) \cdot \frac{H}{2} = (1 - (1 - 1/L)^L) \cdot \frac{H}{2} \geq \frac{H}{4}.$$

In the NO instance case, we have

$$\kappa_{k,\tau} \leq c/wb \quad \text{where} \quad w = 2^{\log(n)^{1-o(1)}} = \Omega(1),$$

then the value of $s_{\text{init}}$ at the end of any stage $k \in [K]$, epoch $\tau \in [n]$ is at most

$$V^*_{t(k,\tau),1}(s_{\text{init}}) = (1 - (1 - \kappa_{k,\tau})^L) \cdot H/2 \leq (1 - (1 - 1/wL)^L) \cdot \frac{H}{2} \leq \frac{1}{w} \cdot \frac{H}{2} \leq \frac{H}{100}.$$

Now we bound the amortized runtime. By Theorem 5, assuming SETH, the total runtime of any NSRL algorithm should be at least $n^{2-o(1)}$, and therefore, the amortized runtime per update should be at least $n^{2-o(1)}/T = (SAH)^{1-o(1)} \cdot T^{-o(1)} \approx (SAH)^{1-o(1)}$ when $T = \text{poly}(SAH)$. This completes the proof. ■

Finally, we leave remarks on the generality of our hardness results.

**Remark 11 (Hardness for maintaining value functions or policies)** *The statement of Theorem 1 asserts the decision version of NSRL requires $(SAH)^{1-o(1)}$ time per update. The same lower bound translates directly to the task of maintaining an approximate $V$-value or maintaining an approximately optimal policy.*

**Remark 12 (Hardness for restricted changes)** *In our construction, we allow the change for both reward and transition kernel, for all states, but indeed, this can be relaxed. For example, similar complexity results follow rather easily if only the reward changes or if only the transition changes. In fact, the lower bound also holds if the changes are restricted to occur on a tiny fraction of the states.*

## 4. Incremental action changes

When the MDP changes only through the introduction of new actions, then we can maintain an $\epsilon$-approximation to value with amortized runtime that depends, polynomially, only on $H$ and $\frac{1}{\epsilon}$ (and not $S$).

**Theorem 13 (Efficient algorithm, incremental changes)** *There is an algorithm with amortized runtime $\widetilde{O}(H^5/\epsilon^3)$ per update that maintains an $\epsilon$-approximation of the value over any sequence of $T$ insertions of actions.*

The approach is given as Algorithm 1. It combines the classic $Q$-value iteration with lazy updates on $V$-value. For each new state-action pair $(s_h, a_h)$, it constructs the empirical transition kernel using samples from $P_h(s_h, a_h)$. The newly added action could potentially affect the state value, and our algorithm propagates the change — lazily — to downstream states. That is, a change to $V$-value is triggered only if it significantly exceeds the previous estimate. The key mathematical intuition is the monotonicity of $V$-value under incremental action changes. The amortized runtime of Algorithm 1 is bounded because the $Q$-value of each state-action is updated rarely, at most $\widetilde{O}(H^3/\epsilon^2 \cdot H^2/\epsilon) = \widetilde{O}(H^5/\epsilon^3)$ times, due to the sparsity of the empirical transition kernel and the lazy updates. The correctness of our algorithm follows from the standard Bernstein type bound and a robust analysis of $Q$-value iteration. The detailed proof can be found at Appendix C.

**Algorithm 1** Lazy updated $Q$-value iteration (Lazy-QVI)

Initialize $N \leftarrow H^3 \log^3(SHT)/\epsilon^2$, $\widehat{V}_h(s_h) \leftarrow 0, \widetilde{V}_h(s_h) \leftarrow 0, \forall s_h \in \mathcal{S}_h, h \in [H]$
**procedure** INSERT($s_h, a_h$)
  Generate $N$ samples $\{\widehat{s}_{h+1,1}, \ldots, \widehat{s}_{h+1,N}\}$ from $P_h(s_h, a_h)$ and reward $r_h(s_h, a_h)$
  $\widehat{P}_h(s_h, a_h) \leftarrow \text{unif}\{\widehat{s}_{h+1,1}, \ldots, \widehat{s}_{h+1,N}\}$
  Call PROPAGATE
**end procedure**
**procedure** PROPAGATE
  **for** $h = H, H-1, \ldots, 1$ **do**
    **for** state-action pair $(s_h, a_h) \in \mathcal{S}_h \times \mathcal{A}_h$ **do** ▷ Update only if there is a change
      $\widehat{Q}_h(s_h, a_h) \leftarrow r_h(s_h, a_h) + \mathbb{E}_{s_{h+1} \sim \widehat{P}_h(s_h, a_h)} \widetilde{V}_{h+1}(s_{h+1})$
      $\widehat{V}_h(s_h) \leftarrow \max_{a_h} \widehat{Q}(s_h, a_h)$
      **If** $\widetilde{V}_h(s_h) \le \widehat{V}_h(s_h) - \epsilon/4H$ **then** $\widetilde{V}_h(s_h) \leftarrow \widehat{V}_h(s_h)$
    **end for**
  **end for**
**end procedure**

Theorem 13 provides an efficient algorithm for approximately optimal policy, one natural question is whether one can maintain the exact optimal policy (or value function) under incremental

action changes. We give a negative answer, showing that $T^{1-o(1)}$ runtime is necessary if one wants to maintain an $O(1/T)$-approximation to the value of optimal policy.

**Theorem 14 (Lower bound, exact optimal policy)** *Unless* SETH *is false, there is a sequence of $T$ action insertions such that no algorithm with amortized runtime $T^{1-o(1)}$ per update can maintain an $O(1/T)$-approximation to the value of optimal policy.*

## 5. Discussion

Ideally, a complexity result should give some hints on the kind of new algorithms that will bypass it. Our result seems to suggest that a successful heuristic approach to NSRL may be one that alternates between additional exploration after each change in parameters and, when this brings diminishing benefits, a restart from scratch. This is not unlike some of the approaches taken by some state-of-the-art applications (Padakandla, 2021). By further developing this and similar approaches, the current challenge of NSRL may be eventually tamed. We also note that our negative result leaves open the NSRL problem in the case of function approximation (Jin et al., 2020; Agarwal et al., 2019); we conjecture that a similar negative result can be proved for this case as well.

## Acknowledgments

The research is supported by NSF CCF-1703925, IIS-1838154, CCF-2106429, CCF-2107187, CCF-1763970, AF2212233, COLL2134095, COLL2212745

## References

Amir Abboud and Virginia Vassilevska Williams. Popular conjectures imply strong lower bounds for dynamic problems. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pages 434–443. IEEE, 2014.

Amir Abboud, Aviad Rubinstein, and Ryan Williams. Distributed pcp theorems for hardness of approximation in p. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 25–36. IEEE, 2017.

Amir Abboud, Raghavendra Addanki, Fabrizio Grandoni, Debmalya Panigrahi, and Barna Saha. Dynamic set cover: improved algorithms and lower bounds. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 114–125, 2019.

Alekh Agarwal, Nan Jiang, Sham M Kakade, and Wen Sun. Reinforcement learning: Theory and algorithms. *CS Dept., UW Seattle, Seattle, WA, USA, Tech. Rep*, pages 10–4, 2019.

Josh Alman, Timothy Chu, Aaron Schild, and Zhao Song. Algorithms and hardness for linear algebra on geometric graphs. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 541–552. IEEE, 2020.

Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

Peter Auer, Thomas Jaksch, and Ronald Ortner. Near-optimal regret bounds for reinforcement learning. *Advances in neural information processing systems*, 21, 2008.

Mohammad Azar, Rémi Munos, and Hilbert J Kappen. Minimax pac bounds on the sample complexity of reinforcement learning with a generative model. *Machine learning*, 91:325–349, 2013.

Arturs Backurs and Piotr Indyk. Edit distance cannot be computed in strongly subquadratic time (unless seth is false). In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 51–58, 2015.

Arturs Backurs, Liam Roditty, Gilad Segal, Virginia Vassilevska Williams, and Nicole Wein. Towards tight approximation bounds for graph diameter and eccentricities. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 267–280, 2018.

Taposh Banerjee, Miao Liu, and Jonathan P How. Quickest change detection approach to optimal control in markov decision processes with model changes. In *2017 American control conference (ACC)*, pages 399–405. IEEE, 2017.

Richard E Bellman. *Dynamic programming*. Princeton university press, 1957.

Dimitri Bertsekas. *Dynamic programming and optimal control: Volume I*, volume 1. Athena scientific, 2012.

Moses Charikar and Paris Siminelakis. Hashing-based-estimators for kernel density in high dimensions. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1032–1043. IEEE, 2017.

Lijie Chen. On the hardness of approximate and exact (bichromatic) maximum inner product. *Theory of Computing*, 16(4):1–50, 2020.

Xi Chen, Christos Papadimitriou, and Binghui Peng. Memory bounds for continual learning. In *2022 IEEE 63th Annual Symposium on Foundations of Computer Science (FOCS)*, 2022.

Wang Chi Cheung, David Simchi-Levi, and Ruihao Zhu. Reinforcement learning for non-stationary markov decision processes: The blessing of (more) optimism. In *International Conference on Machine Learning*, pages 1843–1854. PMLR, 2020.

Bruno C Da Silva, Eduardo W Basso, Ana LC Bazzan, and Paulo M Engel. Dealing with non-stationary environments using context detection. In *Proceedings of the 23rd international conference on Machine learning*, pages 217–224, 2006.

Mina Dalirrooyfard, Ray Li, and Virginia Vassilevska Williams. Hardness of approximate diameter: Now for undirected graphs. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1021–1032. IEEE, 2022.

Travis Dick, Andras Gyorgy, and Csaba Szepesvari. Online learning in markov decision processes with changing cost sequences. In *International Conference on Machine Learning*, pages 512–520. PMLR, 2014.

Omar Darwiche Domingues, Pierre Ménard, Matteo Pirotta, Emilie Kaufmann, and Michal Valko. A kernel-based approach to non-stationary reinforcement learning in metric spaces. In *International Conference on Artificial Intelligence and Statistics*, pages 3538–3546. PMLR, 2021.

Songtao Feng, Ming Yin, Ruiquan Huang, Yu-Xiang Wang, Jing Yang, and Yingbin Liang. Non-stationary reinforcement learning under general function approximation. *arXiv preprint arXiv:2306.00861*, 2023.

Dylan J Foster, Sham M Kakade, Jian Qian, and Alexander Rakhlin. The statistical complexity of interactive decision making. *arXiv preprint arXiv:2112.13487*, 2021.

Ronald A Howard. Dynamic programming and markov processes. 1960.

Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-sat. *Journal of Computer and System Sciences*, 62(2):367–375, 2001.

Haozhe Jiang, Qiwen Cui, Zhihan Xiong, Maryam Fazel, and Simon S Du. A black-box approach for non-stationary multi-agent reinforcement learning. *arXiv preprint arXiv:2306.07465*, 2023.

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

Chi Jin, Zhuoran Yang, Zhaoran Wang, and Michael I Jordan. Provably efficient reinforcement learning with linear function approximation. In *Conference on Learning Theory*, pages 2137–2143. PMLR, 2020.

Khimya Khetarpal, Matthew Riemer, Irina Rish, and Doina Precup. Towards continual reinforcement learning: A review and perspectives. *Journal of Artificial Intelligence Research*, 75:1401–1476, 2022.

Yin Tat Lee and Aaron Sidford. Path finding methods for linear programming: Solving linear programs in $o(\sqrt{rank})$ iterations and faster algorithms for maximum flow. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pages 424–433. IEEE, 2014.

Gen Li, Yuting Wei, Yuejie Chi, Yuantao Gu, and Yuxin Chen. Breaking the sample size barrier in model-based reinforcement learning with a generative model. *Advances in neural information processing systems*, 33:12861–12872, 2020.

Ray Li. Settling seth vs. approximate sparse directed unweighted diameter (up to (nu) nseth). In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1684–1696, 2021.

Yingying Li and Na Li. Online learning for markov decision processes in nonstationary environments: A dynamic regret analysis. In *2019 American Control Conference (ACC)*, pages 1232–1237. IEEE, 2019.

ML Littman. On the complexity of solving markov decision problems. *Proceedings of the 11th International Comference on Uncertainty in Artificial Intelligence*, 1995.

Weichao Mao, Kaiqing Zhang, Ruihao Zhu, David Simchi-Levi, and Tamer Basar. Near-optimal model-free reinforcement learning in non-stationary episodic mdps. In *International Conference on Machine Learning*, pages 7447–7458. PMLR, 2021.

Melkior Ornik and Ufuk Topcu. Learning and planning for time-varying mdps using maximum likelihood estimation. *The Journal of Machine Learning Research*, 22(1):1656–1695, 2021.

Ronald Ortner, Pratik Gajane, and Peter Auer. Variational regret bounds for reinforcement learning. In *Uncertainty in Artificial Intelligence*, pages 81–90. PMLR, 2020.

Sindhu Padakandla. A survey of reinforcement learning algorithms for dynamically varying environments. *ACM Computing Surveys (CSUR)*, 54(6):1–25, 2021.

Sindhu Padakandla, Prabuchandran KJ, and Shalabh Bhatnagar. Reinforcement learning algorithm for non-stationary environments. *Applied Intelligence*, 50:3590–3606, 2020.

Christos H Papadimitriou. Computational complexity. In *Encyclopedia of computer science*, pages 260–265. 2003.

Christos H Papadimitriou and Kenneth Steiglitz. *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1998.

Christos H Papadimitriou and John N Tsitsiklis. The complexity of markov decision processes. *Mathematics of operations research*, 12(3):441–450, 1987.

Binghui Peng. Dynamic influence maximization. *Advances in Neural Information Processing Systems*, 34:10718–10731, 2021.

Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Liam Roditty and Virginia Vassilevska Williams. Fast approximation algorithms for the diameter and radius of sparse graphs. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 515–524, 2013.

Aviad Rubinstein. Hardness of approximate nearest neighbor search. In *Proceedings of the 50th annual ACM SIGACT symposium on theory of computing*, pages 1260–1268, 2018.

Aviad Rubinstein and Virginia Vassilevska Williams. Seth vs approximation. *ACM SIGACT News*, 50(4):57–76, 2019.

Bruno Scherrer. Improved and generalized upper bounds on the complexity of policy iteration. *Advances in Neural Information Processing Systems*, 26, 2013.

Aaron Sidford, Mengdi Wang, Xian Wu, Lin Yang, and Yinyu Ye. Near-optimal time and sample complexities for solving markov decision processes with a generative model. *Advances in Neural Information Processing Systems*, 31, 2018a.

Aaron Sidford, Mengdi Wang, Xian Wu, and Yinyu Ye. Variance reduced value iteration and faster algorithms for solving markov decision processes. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 770–787. SIAM, 2018b.

Aaron Sidford, Mengdi Wang, Lin Yang, and Yinyu Ye. Solving discounted stochastic two-player games with near-optimal time and sample complexity. In *International Conference on Artificial Intelligence and Statistics*, pages 2992–3002. PMLR, 2020.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Ahmed Touati and Pascal Vincent. Efficient learning in non-stationary linear markov decision processes. *arXiv preprint arXiv:2010.12870*, 2020.

Paul Tseng. Solving h-horizon, stationary markov decision problems in time proportional to log (h). *Operations Research Letters*, 9(5):287–297, 1990.

Jan Van Den Brand, Yin Tat Lee, Yang P Liu, Thatchaphol Saranurak, Aaron Sidford, Zhao Song, and Di Wang. Minimum cost flows, mdps, and $\ell_1$-regression in nearly linear time for dense instances. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 859–869, 2021.

Chen-Yu Wei and Haipeng Luo. Non-stationary reinforcement learning without prior knowledge: An optimal black-box approach. In *Conference on Learning Theory*, pages 4300–4354. PMLR, 2021.

Virginia Vassilevska Williams. On some fine-grained questions in algorithms and complexity. In *Proceedings of the international congress of mathematicians: Rio de janeiro 2018*, pages 3447–3487. World Scientific, 2018.

Yinyu Ye. A new complexity result on solving the markov decision problem. *Mathematics of Operations Research*, 30(3):733–749, 2005.

Yinyu Ye. The simplex and policy-iteration methods are strongly polynomial for the markov decision problem with a fixed discount rate. *Mathematics of Operations Research*, 36(4):593–603, 2011.

Huozhi Zhou, Jinglin Chen, Lav R Varshney, and Ashish Jagmohan. Nonstationary reinforcement learning with linear function approximation. *Transactions on Machine Learning Research*, 2022.

## Appendix A. Additional related work

**Computational complexity of reinforcement learning** The computational complexity of (stationary) MDP has been a central topic across multiple disciplines. The study of MDP dates back to Bellman (1957) in 1950s, and since then, there is a long line of work concerning the computational efficiency of MDP (Tseng, 1990; Littman, 1995; Howard, 1960; Ye, 2011; Scherrer, 2013; Ye, 2005; Sidford et al., 2018b,a, 2020; Lee and Sidford, 2014; Van Den Brand et al., 2021; Papadimitriou and Tsitsiklis, 1987). The classical approaches include value iteration, policy iteration and linear programming, see Puterman (2014); Bertsekas (2012) for reference. For a finite horizontal MDP with $S$ states, $A$ actions and $H$ steps, the value iteration could return the optimal policy with runtime $O(S^2AH)$ that is linear in the input size. If the algorithm could sample from the transition function (a.k.a. the generative model), then Azar et al. (2013) provide an algorithm that returns an $\epsilon$-approximation to the $V$-value with runtime $\widetilde{O}(SAH^3/\epsilon^2)$. For non-stationary MDP, it implies an algorithm with runtime $\widetilde{O}(S^2AH + SAH^3T/\epsilon^2)$ for $\epsilon$-value approximation over a sequence of $T$ updates. This is because the algorithm could always re-compute from scratch, and it can sample the transition function in $O(\log(S))$ time using a binary tree data structure, after reading the input initially.

Besides computation complexity, a large number of work concern about the sample complexity in generative model (e.g. Azar et al. (2013); Li et al. (2020)) and regret in model-free RL (e.g. Jin et al. (2018)), in tabular setting as well as functional approximation setting (Foster et al., 2021).

## Appendix B. Missing proof from Section 3

**Proof** [Proof of Lemma 6] This is quite obvious, as the terminal state always stays on itself (Eq. (2)), the reward of $s^{\mathsf{t}}_{h,2}$ is always 0, while the reward of $s^{\mathsf{t}}_{h,1}$ is 0 in phase one and 1 in phase two (Eq. (1)). ■

**Proof** [Proof of Lemma 7] For an element $u \in C_\tau$, a policy could choose to never leave $s^{\mathsf{e}}_u$ (Eq. (3)), and it receives the maximum $H/2$ reward (see Eq. (9)(1)). For an element $u \notin C_\tau$, the policy needs to stay at $s^{\mathsf{e}}_u$ until the end of layer $\ell$ (see Eq. (3)). While at the end of layer $\ell$, it could move to pivotal state $s^{\mathsf{p}}_{h(\ell)+1}$ or stay on itself (Eq. (4)). The later obtains strictly less reward, because the pivotal state could always stay on itself (Eq. (6)), and the value $V^*_{t(k,\tau),H/2}(s^{\mathsf{e}}_{H/2,u}) = 0$ at the end of phase 1 (see Eq. (9)(1)). This completes the proof. ■

**Proof** [Proof of Lemma 8] The $Q$-value of choosing action $a^{\mathsf{b}}_j$ ($j \in [A/2]$) equals

$$
\begin{aligned}
& Q^*_{t(k,\tau),h(\ell,g)-1}(s^{\mathsf{b}}_{h(\ell,g)-1,i}, a^{\mathsf{b}}_j) \\
= & \sum_{u\in[m]} \Pr[s_{h(\ell,g)} = s^{\mathsf{e}}_{h(\ell,g),u}] \cdot V^*_{t(k,\tau),h(\ell,g)}(s^{\mathsf{b}}_{h(\ell,g),u}) \\
= & \sum_{u\in C_\tau} \Pr[s_{h(\ell,g)} = s^{\mathsf{e}}_{h(\ell,g),u}] \cdot \frac{H}{2} + \sum_{u\in[m]\setminus C_\tau} \Pr[s_{h(\ell,g)} = s^{\mathsf{e}}_{h(\ell,g),u}] \cdot V^*_{t(k,\tau),h(\ell)+1}(s^{\mathsf{p}}_{h(\ell)+1}) \\
= & \frac{|C_\tau \cap B_{k,N(g,i,j)}|}{b} \cdot \frac{H}{2} + \left(1 - \frac{|C_\tau \cap B_{k,N(g,i,j)}|}{b}\right) \cdot V^*_{t(k,\tau),\ell+1}(s^{\mathsf{p}}_{h(\ell)+1}).
\end{aligned}
$$

The first step follows from Bellman's equation and the state-action pair $(s^{\mathsf{b}}_{h(\ell,g)-1,i}, a^{\mathsf{b}}_j)$ receives 0 reward (Eq. (1)), the second step follows from Lemma 7, the last step follows from Eq. (5). The proof follows by taking the maximum over action $\{a^{\mathsf{b}}_j\}_{j\in[A/2]}$. ■

**Proof** [Proof of Lemma 9] For each level $\ell \in [L]$, the transition functions of the pivotal state and routing states guarantee that a policy can visit exactly one set state in $\mathcal{H}_\ell$. To see this, it can visit at most one set state because the element state stays on itself till the end of the layer (Eq. (3)). Meanwhile, it can go to any set state $\nu \in [N]$ with $\nu = N(g,i,j)$ for some $i \in [S/2]$ and $g \in [G]$, because it can first go to the pivotal state $s^{\mathsf{p}}_{h(\ell,g-1)+1,0}$ at the beginning of group $g$, then move to $s^{\mathsf{r}}_{h(\ell,g)-1,i}$ through routing states (see Eq. (7)(8)). Combining Lemma 8, we have

$$
\begin{aligned}
&V^*_{t(k,\tau),h(\ell-1)+1}(s^{\mathsf{p}}_{h(\ell-1)+1}) \\
&= \max_{g\in[G]} \max_{i\in[S/4]} V^*_{t(k,\tau),h(\ell,g)-1}(s^{\mathsf{b}}_{h(\ell,g)-1,i}) \\
&= \max_{g\in[G]} \max_{i\in[S/4]} \max_{j\in[A/2]} \left\{ \frac{|C_\tau \cap B_{k,N(g,i,j)}|}{b} \cdot \frac{H}{2} + \left(1 - \frac{|C_\tau \cap B_{k,N(g,i,j)}|}{b}\right) \cdot V^*_{t(k,\tau),h(\ell)+1}(s^{\mathsf{p}}_{h(\ell)+1}) \right\} \\
&= \max_{\nu\in[N]} \left\{ \frac{|C_\tau \cap B_{k,\nu}|}{b} \cdot \frac{H}{2} + \left(1 - \frac{|C_\tau \cap B_{k,\nu}|}{b}\right) \cdot V^*_{t(k,\tau),h(\ell)+1}(s^{\mathsf{p}}_{h(\ell)+1}) \right\}.
\end{aligned}
$$

This completes the proof of the lemma. ■

**Proof** [Proof of Lemma 10] By Lemma 9, for any $\ell \in [L]$, we have

$$
\begin{aligned}
&V^*_{t(k,\tau),h(\ell-1)+1}(s^{\mathsf{p}}_{h(\ell-1)+1}) \\
&= \max_{\nu\in[N]} \left\{ \frac{|C_\tau \cap B_{k,\nu}|}{b} \cdot \frac{H}{2} + \left(1 - \frac{|C_\tau \cap B_{k,\nu}|}{b}\right) \cdot V^*_{t(k,\tau),h(\ell)+1}(s^{\mathsf{p}}_{h(\ell)+1}) \right\} \\
&= \kappa_{k,\tau} \cdot \frac{H}{2} + (1-\kappa_{k,\tau}) V^*_{t(k,\tau),h(\ell)+1}(s^{\mathsf{p}}_{h(\ell)+1}).
\end{aligned}
$$

Solving the above recursion, one has

$$
V^*_{t(k,\tau),1}(s_{\text{init}}) = V^*_{t(k,\tau),1}(s^{\mathsf{p}}_1) = \sum_{\ell=1}^{L} \kappa_{k,\tau}(1-\kappa_{k,\tau})^{\ell-1} \cdot \frac{H}{2} = (1-(1-\kappa_{k,\tau})^L) \cdot \frac{H}{2}.
$$

This completes the proof of the lemma. ■

## Appendix C. Missing proof from Section 4

We first state the concentration bounds used in the paper.

**Lemma 15 (Hoeffding bound)** *Let $X_1, \cdots, X_n$ be $n$ independent bounded variables in $[a_i, b_i]$. Let $X = \sum_{i=1}^n X_i$, then we have*

$$
\Pr[|X - \mathbb{E}[X]| \geq t] \leq 2\exp\left(-\frac{2t^2}{\sum_{i=1}^n (b_i - a_i)^2}\right).
$$

**Lemma 16 (Bernstein bound)** *Let $X_1, \cdots, X_n$ be $n$ independent zero mean random variables and $|X_i| \le M$. Let $X = \sum_{i=1}^n X_i$, $\sigma = \sum_{i=1}^n \mathbb{E}[X_i^2]$ then we have*

$$\Pr[|X| \ge t] \le 2\exp\left(-\frac{2t^2}{Mt/3 + \sigma^2}\right).$$

*In particular, with probability at least $1-\delta$, one has*

$$|X| \le (M/3 + \sigma) \cdot \log(1/\delta).$$

We prove Algorithm 1 gives $\epsilon$-approximation to both $V$-value and $Q$-value. For notation convenience, we drop the subscript of round number $t$ in the proof.

**Lemma 17 (Value approximation)** *At the end of $t$-th update ($t \in [T]$), for any step $h \in [H]$, state $s_h \in \mathcal{S}_h$ and action $a_h$, with probability at least $1 - (SHT)^{-\omega(1)}$, we have*

$$|V_h^*(s_h) - \widehat{V}_h(s_h)| \le \epsilon/2 \quad \textit{and} \quad |Q_h^*(s_h, a_h) - \widehat{Q}_h(s_h, a_h)| \le \epsilon$$

**Proof** We prove the claim by induction. The base case of $h = H$ holds trivially, as there is no error. Suppose the claim holds up to step $h+1$, then for the $h$-th step, we have

$$\begin{aligned}
\widehat{Q}_h(s_h, a_h) &= r_h(s_h, a_h) + \mathop{\mathbb{E}}_{s_{h+1} \sim \widehat{P}_h(s_h, a_h)} \widetilde{V}_{h+1}(s_{h+1}) \\
&= r_h(s_h, a_h) + \mathop{\mathbb{E}}_{s_{h+1} \sim \widehat{P}_h(s_h, a_h)} \widehat{V}_{h+1}(s_{h+1}) \pm \frac{\epsilon}{4H} \\
&= r_h(s_h, a_h) + \mathop{\mathbb{E}}_{s_{h+1} \sim \widehat{P}_h(s_h, a_h)} [V_{h+1}^*(s_{h+1})] \\
&\quad + \mathop{\mathbb{E}}_{s_{h+1} \sim \widehat{P}_h(s_h, a_h)} [\widehat{V}_{h+1}(s_{h+1}) - V_{h+1}^*(s_{h+1})] \pm \frac{\epsilon}{4H}, \qquad (10)
\end{aligned}$$

where the first step follows from the update rule of Algorithm 1, the second step holds since that the propagate value $\widetilde{V}_{h+1}(s_{h+1})$ satisfies

$$\left|\widehat{V}_{h+1}(s_{h+1}) - \widetilde{V}_{h+1}(s_{h+1})\right| \le \frac{\epsilon}{4H}, \quad \forall s_{h+1} \in \mathcal{S}_{h+1}.$$

We bound the second term of Eq. (10) in terms of variance. Define

$$\sigma_h(s_h, a_h)^2 := \mathop{\mathbb{E}}_{s_{h+1} \sim P_h(s_h, a_h)} [V_{h+1}^*(s_{h+1})^2] - \left(\mathop{\mathbb{E}}_{s_{h+1} \sim P_h(s_h, a_h)} [V_{h+1}^*(s_{h+1})]\right)^2.$$

By Bernstein inequality, we have with probability at least $1 - (SHT)^{-\omega(1)}$,

$$\begin{aligned}
\left|\mathop{\mathbb{E}}_{s_{h+1} \sim \widehat{P}_h(s_h, a_h)} [V_{h+1}^*(s_{h+1})] - \mathop{\mathbb{E}}_{s_{h+1} \sim P_h(s_h, a_h)} [V_{h+1}^*(s_{h+1})]\right| &\lesssim \frac{H + \sqrt{N}\sigma_h(s_h, a_h)}{N} \cdot \log(SHT) \\
&\le \frac{\epsilon^2}{H^2} + \frac{\epsilon}{16H^{3/2}} \cdot \sigma_h(s_h, a_h).
\end{aligned}$$

Plugging into Eq. (10), we have

$$\begin{aligned}
&\widehat{Q}_h(s_h,a_h)\\
&= r_h(s_h,a_h) + \mathop{\mathbb{E}}_{s_{h+1}\sim P_h(s_h,a_h)}[V^*_{h+1}(s_{h+1})] + \mathop{\mathbb{E}}_{s_{h+1}\sim \widehat{P}_h(s_h,a_h)}[\widehat{V}_{h+1}(s_{h+1}) - V^*_{h+1}(s_{h+1})]\\
&\quad \pm \frac{\epsilon}{16H^{3/2}}\cdot\sigma_h(s_h,a_h) \pm \frac{\epsilon}{3H}\\
&= Q^*_h(s_h,a_h) + \mathop{\mathbb{E}}_{s_{h+1}\sim \widehat{P}_h(s_h,a_h)}[\widehat{V}_{h+1}(s_{h+1}) - V^*_{h+1}(s_{h+1})] \pm \frac{\epsilon}{16H^{3/2}}\cdot\sigma_h(s_h,a_h) \pm \frac{\epsilon}{3H}. \quad (11)
\end{aligned}$$

We bound the $V$-value difference $\widehat{V}_h(s_h) - V^*_h(s_h)$ and provide upper and lower bounds separately.

**Upper bound $\widehat{V}_h(s_h) - V^*_h(s_h)$.** Let $\widehat{\pi}$ be the policy induced by $\widehat{Q}$, that is, for any state $s_\ell \in \mathcal{S}_\ell$, $\widehat{\pi}(s_\ell) = \mathsf{argmax}_{a_\ell}\widehat{Q}_\ell(s_\ell,a_\ell)$. Then for any state $s_h \in \mathcal{S}_h$, one has

$$\begin{aligned}
&\widehat{V}_h(s_h) - V^*_h(s_h)\\
&= \widehat{Q}_h(s_h,\widehat{\pi}(s_h)) - Q^*_h(s_h,\pi^*(s_h))\\
&= \widehat{Q}_h(s_h,\widehat{\pi}(s_h)) - Q^*_h(s_h,\widehat{\pi}(s_h)) + Q^*_h(s_h,\widehat{\pi}(s_h)) - Q^*_h(s_h,\pi^*(s_h))\\
&\le \widehat{Q}_h(s_h,\widehat{\pi}(s_h)) - Q^*_h(s_h,\widehat{\pi}(s_h))\\
&\le \mathop{\mathbb{E}}_{s_{h+1}\sim \widehat{P}_h(s_h,\widehat{\pi}(s_h))}[\widehat{V}_{h+1}(s_{h+1}) - V^*_{h+1}(s_{h+1})] \pm \frac{\epsilon}{16H^{3/2}}\sigma_h(s_h,\widehat{\pi}(s_h)) + \frac{\epsilon}{3H}, \quad (12)
\end{aligned}$$

where the third step follows from the optimality of $\pi^*$, the fourth step follows from Eq. (11).

Fix the state $s_h \in \mathcal{S}_h$, for any step $\ell \in [h:H]$ and state $s_\ell \in \mathcal{S}_\ell$, let $\widehat{p}(s_\ell)$ be the probability that policy $\hat{\pi}$ goes to state $s_\ell$, starting from $s_h$. Recurring Eq. (12), we obtain

$$\begin{aligned}
\widehat{V}_h(s_h) - V^*_h(s_h) &\le \frac{\epsilon}{16H^{3/2}}\cdot\sum_{\ell=h}^{H}\sum_{s_\ell\in\mathcal{S}_\ell}\widehat{p}(s_\ell)\sigma_\ell(s_\ell,\widehat{\pi}(s_\ell)) + \frac{\epsilon}{3}\\
&\le \frac{\epsilon}{16H}\sqrt{\sum_{\ell=h}^{H}\widehat{p}(s_\ell)\sigma_\ell(s_\ell,\widehat{\pi}(s_\ell))^2} + \frac{\epsilon}{3}. \quad (13)
\end{aligned}$$

Here the first step follows Eq. (12), the second step follows from Cauchy Schwarz inequality and $\sum_{s_\ell\in\mathcal{S}_\ell}\widehat{p}(s_\ell) = 1$ holds for any $\ell \ge h$.

We need the following two technical Lemmas.

**Lemma 18 (Connection with empirical variance)** *Define the empirical variance*

$$\widehat{\sigma}_h(s_h,a_h)^2 := \mathop{\mathbb{E}}_{s_{h+1}\sim\widehat{P}_h(s_h,a_h)}[\widehat{V}_{h+1}(s_{h+1})^2] - \left(\mathop{\mathbb{E}}_{s_{h+1}\sim\widehat{P}_h(s_h,a_h)}[\widehat{V}_{h+1}(s_{h+1})]\right)^2$$

*Then with probability at least* $1-(SHT)^{-\omega(1)}$*, one has*

$$|\sigma_h(s_h,a_h)^2 - \widehat{\sigma}_h(s_h,a_h)^2| \le H.$$

**Proof** First, by Hoeffding inequality, with probability at least $1-(SHT)^{-\omega(1)}$, one has

$$\left|\mathop{\mathbb{E}}_{s_{h+1}\sim\widehat{P}_h(s_h,a_h)}[\widehat{V}_{h+1}(s_{h+1})^2]-\mathop{\mathbb{E}}_{s_{h+1}\sim P_h(s_h,a_h)}[\widehat{V}_{h+1}(s_{h+1})^2]\right|\le\frac{4H^2\sqrt{N}\log(SHT)}{N}\le H/4.$$

By induction hypothesis, one has $|\widehat{V}_{h+1}(s_{h+1})-V_{h+1}(s_{h+1})|\le\epsilon$ for any state $s_{h+1}\in\mathcal{S}_{h+1}$, and therefore,

$$\left|\mathop{\mathbb{E}}_{s_{h+1}\sim P_h(s_h,a_h)}[\widehat{V}_{h+1}(s_{h+1})^2-V_{h+1}(s_{h+1})^2]\right|\le 2\epsilon H\le H/4$$

Similarly, by Hoeffding bound, we have with probability at least $1-(SHT)^{-\omega(1)}$,

$$\left|\mathop{\mathbb{E}}_{s_{h+1}\sim\widehat{P}_h(s_h,a_h)}[\widehat{V}_{h+1}(s_{h+1})]-\mathop{\mathbb{E}}_{s_{h+1}\sim P_h(s_h,a_h)}[\widehat{V}_{h+1}(s_{h+1})]\right|\le\frac{4\sqrt{N}H\log(SHT)}{N}\le\epsilon$$

and by induction hypothesis,

$$\left|\mathop{\mathbb{E}}_{s_{h+1}\sim P_h(s_h,a_h)}[\widehat{V}_{h+1}(s_{h+1})-V_{h+1}(s_{h+1})]\right|\le\epsilon$$

Combining the above four inequalities, one can conclude the proof. ■

**Lemma 19 (Upper bound on empirical variance)** *We have*

$$\sum_{\ell=h}^{H}\widehat{p}(s_\ell)\widehat{\sigma}_\ell(s_\ell,\widehat{\pi}(s_\ell))^2\le 3H^2$$

**Proof** We have

$$\begin{aligned}
&\sum_{\ell=h}^{H}\sum_{s_\ell\in\mathcal{S}_\ell}\widehat{p}(s_\ell)\widehat{\sigma}_\ell(s_\ell,\widehat{\pi}(s_\ell))^2\\
&=\sum_{\ell=h}^{H}\sum_{s_\ell\in\mathcal{S}_\ell}\widehat{p}(s_\ell)\cdot\left(\mathop{\mathbb{E}}_{s_{\ell+1}\sim\widehat{P}_\ell(s_\ell,\widehat{\pi}(s_\ell))}[\widehat{V}_{\ell+1}(s_{\ell+1})^2]-\left(\mathop{\mathbb{E}}_{s_{\ell+1}\sim\widehat{P}_\ell(s_\ell,\widehat{\pi}(s_\ell))}[\widehat{V}_{\ell+1}(s_{\ell+1})]\right)^2\right)\\
&\le\sum_{\ell=h+1}^{H}\sum_{s_\ell\in\mathcal{S}_\ell}\widehat{p}(s_\ell)\left(\widehat{V}_\ell(s_\ell)^2-\left(\mathop{\mathbb{E}}_{s_{\ell+1}\sim\widehat{P}(s_\ell,\widehat{\pi}(s_\ell))}[\widehat{V}_{\ell+1}(s_{\ell+1})]\right)^2\right)+1\\
&=\sum_{\ell=h+1}^{H}\sum_{s_\ell\in\mathcal{S}_\ell}\widehat{p}(s_\ell)\left(\left(\mathop{\mathbb{E}}_{s_{\ell+1}\sim\widehat{P}(s_\ell,\widehat{\pi}(s_\ell))}[\widetilde{V}_{\ell+1}(s_{\ell+1})+r_\ell(s_\ell,\widehat{\pi}(s_\ell))]\right)^2-\left(\mathop{\mathbb{E}}_{s_{\ell+1}\sim\widehat{P}(s_\ell,\widehat{\pi}(s_\ell))}[\widehat{V}_{\ell+1}(s_{\ell+1})]\right)^2\right)+1\\
&\le\sum_{\ell=h+1}^{H}\sum_{s_\ell\in\mathcal{S}_\ell}\widehat{p}(s_\ell)\cdot 2H\cdot(1+\epsilon/H)+1\le 3H^2.
\end{aligned}$$

The first step follows from the definition of empirical variance $\widehat{\sigma}_\ell$. The second step is important and it holds due to the definition of visiting probability $\widehat{p}_\ell$, and we use the naive bound of $\widehat{V}_H(s_H)\le 1$

for any state $s_H \in \mathcal{S}_H$ in last step. The third step holds due to the definition of $\widehat{V}_\ell(s_\ell)$. The last step holds due to

$$\left| \mathop{\mathbb{E}}_{s_{\ell+1}\sim\widehat{P}(s_\ell,\widehat{\pi}(s_\ell))} \widetilde{V}_{\ell+1}(s_{\ell+1}) + r_\ell(s_\ell,\widehat{\pi}(s_\ell)) - \widehat{V}_{\ell+1}(s_{\ell+1}) \right| \le 1 + \epsilon/H$$

as $|\widetilde{V}_{\ell+1}(s_{\ell+1}) - \widehat{V}_{\ell+1}(s_{\ell+1})| \le \epsilon/H$ and $r_\ell(s_\ell,\widehat{\pi}(s_\ell)) \le 1$. ■

Combining Lemma 18, Lemma 19 and Eq. (13), we have that

$$\begin{aligned}
\widehat{V}_h(s_h) - V_h^*(s_h) &\le \frac{\epsilon}{16H}\sqrt{\sum_{\ell=h}^{H} \widehat{p}(s_\ell)\sigma_\ell(s_\ell,\widehat{\pi}(s_\ell))^2} + \frac{\epsilon}{3} \\
&= \frac{\epsilon}{16H}\sqrt{\sum_{\ell=h}^{H} \widehat{p}(s_\ell)\widehat{\sigma}_\ell(s_\ell,\widehat{\pi}(s_\ell))^2 + H^2} + \frac{\epsilon}{3} \\
&\le \frac{\epsilon}{16H}\cdot\sqrt{3H^2+H^2} + \frac{\epsilon}{3} \le \frac{\epsilon}{2}
\end{aligned} \tag{14}$$

**Lower bound $\widehat{V}_h(s_h) - V_h^*(s_h)$.** The proof for lower bound is similar. First, we have

$$\begin{aligned}
&V_h^*(s_h) - \widehat{V}_h(s_h) \\
&= Q_h^*(s_h,\pi^*(s_h)) - \widehat{Q}_h(s_h,\widehat{\pi}(s_h)) \\
&= Q_h^*(s_h,\pi^*(s_h)) - \widehat{Q}_h(s_h,\pi^*(s_h)) + \widehat{Q}_h(s_h,\pi^*(s_h)) - \widehat{Q}_h(s_h,\widehat{\pi}(s_h)) \\
&\le Q_h^*(s_h,\pi^*(s_h)) - \widehat{Q}_h(s_h,\pi^*(s_h)) \\
&\le \mathop{\mathbb{E}}_{s_{h+1}\sim\widehat{P}_h(s_h,\pi^*(s_h))} [V_{h+1}^*(s_{h+1}) - \widehat{V}_{h+1}(s_{h+1})] + \frac{\epsilon}{16H^{3/2}}\sigma_h(s_h,\pi^*(s_h)) + \frac{\epsilon}{3H},
\end{aligned} \tag{15}$$

where the third step follows from the optimality of $\widehat{\pi}$, the fourth step follows from Eq. (11).

Using Hoeffding bound and the induction hypothesis $|V_{h+1}^*(s_{h+1}) - \widehat{V}_{h+1}(s_{h+1})| \le \epsilon$, with probability at least $1 - (SHT)^{-\omega(1)}$, we have

$$\begin{aligned}
&\mathop{\mathbb{E}}_{s_{h+1}\sim\widehat{P}_h(s_h,\pi^*(s_h))} [V_{h+1}^*(s_{h+1}) - \widehat{V}_{h+1}(s_{h+1})] - \mathop{\mathbb{E}}_{s_{h+1}\sim P_h(s_h,\pi^*(s_h))} [V_{h+1}^*(s_{h+1}) - \widehat{V}_{h+1}(s_{h+1})] \\
&\le \frac{2\epsilon\sqrt{N}\log(SHT)}{N} \le \frac{\epsilon}{24H}.
\end{aligned}$$

Plug into Eq. (15), we have

$$\begin{aligned}
V_h^*(s_h) - \widehat{V}_h(s_h) \le &\mathop{\mathbb{E}}_{s_{h+1}\sim P_h(s_h,\pi^*(s_h))} [V_{h+1}^*(s_{h+1}) - \widehat{V}_{h+1}(s_{h+1})] \\
&+ \frac{\epsilon}{16H^{3/2}}\sigma_h(s_h,\pi^*(s_h)) + \frac{3\epsilon}{8H}.
\end{aligned}$$

Fix the state $s_h \in \mathcal{S}_h$, for any step $\ell \in [h:H]$ and state $s_\ell \in \mathcal{S}_\ell$, let $p^*(s_\ell)$ be the probability that policy $\pi^*$ goes to state $s_\ell$, starting from $s_h$. Recurring the above equation, we obtain

$$\begin{aligned} V_h^*(s_h) - \widehat{V}_h(s_h) &\le \frac{\epsilon}{16H^{3/2}} \cdot \sum_{\ell=h}^{H} \sum_{s_\ell \in \mathcal{S}_\ell} p^*(s_\ell)\sigma_\ell(s_\ell, \pi^*(s_\ell)) + \frac{3\epsilon}{8} \\ &\le \frac{\epsilon}{16H} \sqrt{\sum_{\ell=h}^{H} p^*(s_\ell)\sigma_\ell(s_\ell, \pi^*(s_\ell))^2} + \frac{3\epsilon}{8}. \\ &\le \frac{\epsilon}{16H}\sqrt{3H^2} + \frac{\epsilon}{3} \le \epsilon/2. \end{aligned} \tag{16}$$

We use Cauchy Schwarz in the second step, the third step follows from the following Lemma (the proof is similar to Lemma 19 and we omit it here).

**Lemma 20 (Upper bound on variance)** *We have*

$$\sum_{\ell=h}^{H} p^*(s_\ell)\sigma_\ell(s_\ell, \pi^*(s_\ell))^2 \le 3H^2.$$

Combining Eq. (14) and Eq. (16), we conclude the proof for $V$-value.
For $Q$-value, we have with probability at least $1 - (SHT)^{-\omega(1)}$,

$$\begin{aligned} \widehat{Q}_h(s_h, a_h) &= r_h(s_h, a_h) + \mathop{\mathbb{E}}_{s_{h+1} \sim \widehat{P}_h(s_h, a_h)} \widetilde{V}_{h+1}(s_{h+1}) \\ &= r_h(s_h, a_h) + \mathop{\mathbb{E}}_{s_{h+1} \sim \widehat{P}_h(s_h, a_h)} \widehat{V}_{h+1}(s_{h+1}) \pm \frac{\epsilon}{4H} \\ &= r_h(s_h, a_h) + \mathop{\mathbb{E}}_{s_{h+1} \sim \widehat{P}_h(s_h, a_h)} V_{h+1}^*(s_{h+1}) \pm \epsilon/2 \pm \frac{\epsilon}{4H} \\ &= r_h(s_h, a_h) + \mathop{\mathbb{E}}_{s_{h+1} \sim P_h(s_h, a_h)} V_{h+1}^*(s_{h+1}) \pm \epsilon \\ &= Q_h^*(s_h, a_h) \pm \epsilon. \end{aligned}$$

The first step uses the update rule of Algorithm 1, the second step holds since $|\widetilde{V}_{h+1}(s_{h+1}) - \widehat{V}_{h+1}(s_{h+1})| \le \epsilon/4H$. The third follows from the guarantee of $V$-value, and the fourth step follows from Hoeffding bounds and the last step follows from Bellman equation. We finish the induction and complete the proof here. ■

We next bound the total update time of Algorithm 1.

**Lemma 21 (Total update time)** *The total update time of Algorithm 1 is at most $\widetilde{O}(TH^5/\epsilon^3)$ over a sequence of $T$ action insertions.*

**Proof** For each new action $(s_h, a_h)$, the construction of $\widehat{P}_h(s_h, a_h)$ takes $\widetilde{O}(N) = \widetilde{O}(H^3/\epsilon^2)$ time. The major overhead comes from the PROPAGATE part. First, note the propagated $V$-value $\widetilde{V}_h(s_h)$ of a state $s_h$ can change at most $H/(\epsilon/4H) = O(H^2/\epsilon)$ times. Next, for each state-action pair $(s_h, a_h)$, the $Q$-value $\widehat{Q}_h(s_h, a_h) = r_h(s_h, a_h) + \mathbb{E}_{s_{h+1} \sim \widehat{P}_h(s_h, a_h)} \widetilde{V}_{h+1}(s_{h+1})$ can change at most

$O(N \cdot H^2/\epsilon) = \widetilde{O}(H^5/\epsilon^3)$ times, because the support of $\widehat{P}_h(s_h, a_h)$ has size at most $N$, and each estimate $\widetilde{V}_{h+1}(s_{h+1})$ changes at $O(H^2/\epsilon)$ times as stated above. The total number of state-action pair is bounded by $T$. We conclude the proof. ■

The proof of Theorem 13 follows directly from Lemma 17 and Lemma 21.

We next prove the lower bound.

**Proof** [Proof of Theorem 14] Let $n = T^{1-o(1)}$, $m = n^{o(1)}$. We reduce from MAX-IP with sets $B_1, \ldots, B_n$ and $C_1, \ldots, C_n$ defined over ground element $[m]$. The MDP contains $H = 3$ steps. There is one single initial state $s_1$ at the first step $h = 1$. In the second step ($h = 2$), there are $m = n^{o(1)}$ states $s_{2,1}, \ldots, s_{2,m}$, and at the last step ($h = 3$), there are two states $s_{3,1}$, $s_{3,2}$.

The sequence of new actions is as follow. There is one action $a_3$ for the last step, and the reward satisfies $r_3(s_{3,1}, a_3) = 1$ and $r_3(s_{3,1}, a_3) = 0$, i.e., the reward is 1 for $s_{3,1}$ and 0 for $s_{3,2}$. There are $n$ actions $a_{1,1}, \ldots, a_{1,n}$ for the initial state at the first step, and we have

$$P_1(s_1, a_{1,i}) = \mathrm{unif}(\{s_{2,k} : k \in B_i\}) \quad \text{and} \quad r_1(s_1, a_{1,i}) = 0 \quad \forall i \in [n].$$

The rest sequence divides into $n$ epochs, and in the $j$-th epoch ($j \in [n]$), there is one new action $a_{2,j}$ for each state $\{s_{2,k}\}_{k\in[m]}$. Let $\delta = 1/4n$. At the end of $j$-th epoch, $t(j) \in [T]$, the transition and the reward of the new action $a_{2,j}$ satisfies

$$P_2(s_k, a_{2,j}) = \begin{cases} (\frac{j}{n+1} + \delta, 1 - \frac{j}{n+1} - \delta) & k \in C_j \\ (\frac{j}{n+1}, 1 - \frac{j}{n+1}) & k \notin C_j \end{cases} \quad \text{and} \quad r_2(s_k, a_{2,j}) = 0.$$

In summary, the total number of state-action pairs at the end is $2 + n + mn = n^{1+o(1)} = T$.

First, a simple observation on the value function

**Lemma 22** *At the end of epoch $j \in [n]$, the optimal policy satisfies*

- $V^*_{t(j)}(s_{3,1}) = 1$ *and* $V^*_{t(j)}(s_{3,1}) = 0$
- $V^*_{t(j)}(s_{2,k}) = \frac{j}{n+1} + \delta$ *when* $k \in C_j$ *and* $V^*_{t(j)}(s_{2,k}) = \frac{j}{n+1}$ *otherwise*
- $V^*_{t(j)}(s_1) = \frac{j}{n+1} + \kappa_j \cdot \delta$, *where* $\kappa_j = \max_{i\in[n]} \frac{|C_j \cap B_i|}{b}$

**Proof** The first claim is trivial. The second claim holds since the action $a_{2,j}$ is the optimal choice by the end of epoch $j$, and its value equals $Q^*_{t(j)}(s_{2,k}, a_j) = \frac{j}{n+1} + \delta$ when $k \in C_j$ and $Q^*_{t(j)}(s_{2,k}, a_j) = \frac{j}{n+1}$ when $k \notin C_j$. For the last claim, for any $i \in [n]$, we have

$$\begin{aligned} Q^*_{t(j)}(s_1, a_{1,i}) &= \sum_{k\in[m]} \Pr[s_2 = s_{2,k}] \cdot V^*_{t(j)}(s_{2,k}) \\ &= \sum_{k\in C_j} \Pr[s_2 = s_{2,k}] \cdot V^*_{t(j)}(s_{2,k}) + \sum_{k\in[m]\setminus C_j} \Pr[s_2 = s_{2,k}] \cdot V^*_{t(j)}(s_{2,k}) \\ &= \frac{|B_i \cap C_j|}{b} \cdot \left(\frac{j}{n+1} + \delta\right) + \left(1 - \frac{|B_i \cap C_j|}{b}\right) \cdot \frac{j}{n+1} \\ &= \frac{j}{n+1} + \delta \cdot \frac{|B_i \cap C_j|}{b}. \end{aligned}$$

Taking maximum over $i \in [n]$, we have $V^*_{t(j)}(s_1) = \frac{j}{n+1} + \kappa_j \cdot \delta$. ■

Hence, any algorithm that returns $\delta/b = O(1/mn) = O(1/T)$ approximation to optimal $V$-value could distinguish between YES/NO instance of MAX-IP, and therefore, assuming SETH is true, there is no algorithm with $n^{2-o(1)}/T = T^{1-o(1)}$ amortized runtime per update. ■