TOWARDS DEPLOYMENT-EFFICIENT REINFORCEMENT LEARNING: LOWER BOUND AND OPTIMALITY

Jiawei Huang*†, **Jinglin Chen**†, **Li Zhao**‡, **Tao Qin**‡, **Nan Jiang**†, **Tie-Yan Liu**‡
† Department of Computer Science, University of Illinois at Urbana-Champaign
{jiaweih, jinglinc, nanjiang}@illinois.edu
‡ Microsoft Research Asia
{lizo, taoqin, tyliu}@microsoft.com

ABSTRACT

Deployment efficiency is an important criterion for many real-world applications of reinforcement learning (RL). Despite the community's increasing interest, there lacks a formal theoretical formulation for the problem. In this paper, we propose such a formulation for deployment-efficient RL (DE-RL) from an "optimization with constraints" perspective: we are interested in exploring an MDP and obtaining a near-optimal policy within minimal *deployment complexity*, whereas in each deployment the policy can sample a large batch of data. Using finite-horizon linear MDPs as a concrete structural model, we reveal the fundamental limit in achieving deployment efficiency by establishing information-theoretic lower bounds, and provide algorithms that achieve the optimal deployment efficiency. Moreover, our formulation for DE-RL is flexible and can serve as a building block for other practically relevant settings; we give "Safe DE-RL" and "Sample-Efficient DE-RL" as two examples, which may be worth future investigation.

1 INTRODUCTION

In many real-world applications, deploying a new policy to replace the previous one is costly, while generating a large batch of samples with an already deployed policy can be relatively fast and cheap. For example, in recommendation systems (Afsar et al., 2021), education software (Bennane et al., 2013), and healthcare (Yu et al., 2019), the new recommendation, teaching, or medical treatment strategy must pass several internal tests to ensure safety and practicality before being deployed, which can be time-consuming. On the other hand, the algorithm may be able to collect a large amount of samples in a short period of time if the system serves a large population of users. Besides, in robotics applications (Kober et al., 2013), deploying a new policy usually involves operations on the hardware level, which requires non-negligible physical labor and long waiting periods, while sampling trajectories is relatively less laborious. However, deployment efficiency was neglected in most of existing RL literatures. Even for those few works considering this important criterion (Bai et al., 2020; Gao et al., 2021; Matsushima et al., 2021), either their settings or methods have limitations in the scenarios described above, or a formal mathematical formulation is missing. We defer a detailed discussion of these related works to Section 1.1.

In order to close the gap between existing RL settings and real-world applications requiring high deployment efficiency, our first contribution is to provide a formal definition and tractable objective for Deployment-Efficient Reinforcement Learning (DE-RL) via an "optimization with constraints" perspective. Roughly speaking, we are interested in minimizing the number of deployments K under two constraints: (a) after deploying K times, the algorithm can return a near-optimal policy, and (b) the number of *trajectories* collected in each deployment, denoted as N, is at the same level across K deployments, and it can be large but should still be polynomial in standard parameters. Similar to the notion of sample complexity in online RL, we will refer to K as *deployment complexity*.

*Work done during the internship at Microsoft Research Asia.

To provide a more quantitative understanding, we instantiate our DE-RL framework in finite-horizon linear MDPs[1] (Jin et al., 2019) and develop the essential theory. The main questions we address are:

Q1: What is the optimum of the deployment efficiency in our DE-RL setting?
Q2: Can we achieve the optimal deployment efficiency in our DE-RL setting?

When answering these questions, we separately study algorithms with or without being constrained to deploy deterministic policies each time. While deploying more general forms of policies can be practical (e.g., randomized experiments on a population of users can be viewed as deploying a mixture of deterministic policies), most previous theoretical works in related settings exclusively focused on upper and lower bounds for algorithms using deterministic policies (Jin et al., 2019; Wang et al., 2020b; Gao et al., 2021). As we will show, the origin of the difficulty in optimizing deployment efficiency and the principle in algorithm design to achieve optimal deployment efficiency are generally different in these two settings, and therefore, we believe both of them are of independent interests.

As our second contribution, in Section 3, we answer Q1 by providing information-theoretic lower bounds for the required number of deployments under the constraints of (a) and (b) in Def 2.1. We establish $\Omega(dH)$ and $\widetilde{\Omega}(H)$ lower bounds for algorithms with and without the constraints of deploying deterministic policies, respectively. Contrary to the impression given by previous empirical works (Matsushima et al., 2021), even if we can deploy unrestricted policies, the minimal number of deployments cannot be reduced to a constant without additional assumptions, which sheds light on the fundamental limitation in achieving deployment efficiency. Besides, in the line of work on "horizon-free RL" (e.g., Wang et al., 2020a), it is shown that RL problem is not significantly harder than bandits (i.e., when $H = 1$) when we consider sample complexity. In contrast, the H dependence in our lower bound reveals some fundamental hardness that is specific to long-horizon RL, particularly in the deployment-efficient setting. [2] Such hardness results were originally conjectured by Jiang & Agarwal (2018), but no hardness has been shown in sample-complexity settings.

After identifying the limitation of deployment efficiency, as our third contribution, we address Q2 by proposing novel algorithms whose deployment efficiency match the lower bounds. In Section 4.1, we propose an algorithm deploying deterministic policies, which is based on Least-Square Value Iteration with reward bonus (Jin et al., 2019) and a layer-by-layer exploration strategy, and can return an ε-optimal policy within $O(dH)$ deployments. As part of its analysis, we prove Lemma 4.2 as a technical contribution, which can be regarded as a finite-sample version of the well-known "Elliptical Potential Lemma"(Carpentier et al., 2020) and may be of independent interest. Moreover, our analysis based on Lemma 4.2 can be applied to the reward-free setting (Jin et al., 2020; Wang et al., 2020b) and achieve the same optimal deployment efficiency. In Section 4.2, we focus on algorithms which can deploy arbitrary policies. They are much more challenging because it requires us to find a provably exploratory stochastic policy without interacting with the environment. To our knowledge, Agarwal et al. (2020b) is the only work tackling a similar problem, but their algorithm is model-based which relies on a strong assumption about the realizability of the true dynamics and a sampling oracle that allows the agent to sample data from the model, and how to solve the problem in linear MDPs without a model class is still an open problem. To overcome this challenge, we propose a model-free layer-by-layer exploration algorithm based on a novel covariance matrix estimation technique, and prove that it requires $\Theta(H)$ deployments to return an ε-optimal policy, which only differs from the lower bound $\widetilde{\Omega}(H)$ by a logarithmic factor. Although the per-deployment sample complexity of our algorithm has dependence on a "reachability coefficient" (see Def. 4.3), similar quantities also appear in related works (Zanette et al., 2020; Agarwal et al., 2020b; Modi et al., 2021) and we conjecture that it is unavoidable and leave the investigation to future work.

Finally, thanks to the flexibility of our "optimization with constraints" perspective, our DE-RL setting can serve as a building block for more advanced and practically relevant settings where optimizing the number of deployments is an important consideration. In Appendix F, we propose two potentially interesting settings: "Safe DE-RL" and "Sample-Efficient DE-RL", by introducing constraints regarding safety and sample efficiency, respectively.

[1] Although we focus on linear MDPs, the core idea can be extended to more general settings such as RL with general function approximation (Kong et al., 2021).

[2] Although (Wang et al., 2020a) considered stationary MDP, as shown in our Corollary 3.3, the lower bounds of deployment complexity is still related to H.

1.1 Closely Related Works

We defer the detailed discussion of previous literatures about pure online RL and pure offline RL to Appendix A, and mainly focus on those literatures which considered deployment efficiency and more related to us in this section.

To our knowledge, the term "deployment efficiency" was first coined by Matsushima et al. (2021), but they did not provide a concrete mathematical formulation that is amendable to theoretical investigation. In existing theoretical works, low switching cost is a concept closely related to deployment efficiency, and has been studied in both bandit (Esfandiari et al., 2020; Han et al., 2020; Gu et al., 2021; Ruan et al., 2021) and RL settings (Bai et al., 2020; Gao et al., 2021; Kong et al., 2021). Another related concept is concurrent RL, as proposed by Guo & Brunskill (2015). We highlight the difference with them in two-folds from problem setting and techniques.

As for the problem setting, existing literature on low switching cost mainly focuses on sub-linear regret guarantees, which does not directly implies a near-optimal policy after a number of policy deployments[3]. Besides, low switching-cost RL algorithms (Bai et al., 2020; Gao et al., 2021; Kong et al., 2021) rely on adaptive switching strategies (i.e., the interval between policy switching is not fixed), which can be difficult to implement in practical scenarios. For example, in recommendation or education systems, once deployed, a policy usually needs to interact with the population of users for a fair amount of time and generate a lot of data. Moreover, since policy preparation is time-consuming (which is what motivates our work to begin with), it is practically difficult if not impossible to change the policy immediately once collecting enough data for policy update, and it will be a significant overhead compared to a short policy switch interval. Therefore, in applications we target at, it is more reasonable to assume that the sample size in each deployment (i.e., between policy switching) has the same order of magnitude and is large enough so that the overhead of policy preparation can be ignored.

More importantly, on the technical side, previous theoretical works on low switching cost mostly use deterministic policies in each deployment, which is easier to analyze. This issue also applies to the work of Guo & Brunskill (2015) on concurrent PAC RL. However, if the agent can deploy stochastic (and possibly non-Markov) policies (e.g., a mixture of deterministic policies), then intuitively—and as reflected in our lower bounds—exploration can be done much more deployment-efficiently, and we provide a stochastic policy algorithm that achieves an $\widetilde{O}(H)$ deployment complexity and overcomes the $\Omega(dH)$ lower bounds for deterministic policy algorithms (Gao et al., 2021).

2 Preliminaries

Notation Throughout our paper, for $n \in \mathbb{Z}^+$, we will denote $[n] = \{1, 2, ..., n\}$. $\lceil \cdot \rceil$ denotes the ceiling function. Unless otherwise specified, for vector $x \in \mathbb{R}^d$ and matrix $X \in \mathbb{R}^{d\times d}$, $\|x\|$ denotes the vector l_2-norm of x and $\|X\|$ denotes the largest singular value of X. We will use standard big-oh notations $O(\cdot), \Omega(\cdot), \Theta(\cdot)$, and notations such as $\widetilde{O}(\cdot)$ to suppress logarithmic factors.

2.1 Episodic Reinforcement Learning

We consider an episodic Markov Decision Process denoted by $M(\mathcal{S}, \mathcal{A}, H, P, r)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the finite action space, H is the horizon length, and $P = \{P_h\}_{h=1}^H$ and $r = \{r_h\}_{h=1}^H$ denote the transition and the reward functions. At the beginning of each episode, the environment will sample an initial state s_1 from the initial state distribution d_1. Then, for each time step $h \in [H]$, the agent selects an action $a_h \in \mathcal{A}$, interacts with the environment, receives a reward $r_h(s_h, a_h)$, and transitions to the next state s_{h+1}. The episode will terminate once s_{H+1} is reached.

A (Markov) policy $\pi_h(\cdot)$ at step h is a function mapping from $\mathcal{S} \to \Delta(\mathcal{A})$, where $\Delta(\mathcal{A})$ denotes the probability simplex over the action space. With a slight abuse of notation, when $\pi_h(\cdot)$ is a deterministic policy, we will assume $\pi_h(\cdot) : \mathcal{S} \to \mathcal{A}$. A full (Markov) policy $\pi = \{\pi_1, \pi_2, ..., \pi_H\}$ specifies such a mapping for each time step. We use $V_h^\pi(s)$ and $Q_h^\pi(s, a)$ to denote the value function

[3] Although the conversion from sub-linear regret to polynominal sample complexity is possible ("online-to-batch"), we show in Appendix A that to achieve accuracy ε after conversion, the number of deployments of previous low-switching cost algorithms has dependence on ε, whereas our guarantee does not.

and Q-function at step $h \in [H]$, which are defined as:

$$V_h^\pi(s) = \mathbb{E}[\sum_{h'=h}^{H} r_{h'}(s_{h'}, a_{h'})|s_h = s, \pi], \quad Q_h^\pi(s,a) = \mathbb{E}[\sum_{h'=h}^{H} r_{h'}(s_{h'}, a_{h'})|s_h = s, a_h = a, \pi]$$

We also use $V_h^*(\cdot)$ and $Q_h^*(\cdot,\cdot)$ to denote the optimal value functions and use π^* to denote the optimal policy that maximizes the expected return $J(\pi) := \mathbb{E}[\sum_{h=1}^{H} r(s_h, a_h)|\pi]$. In some occasions, we use $V_h^\pi(s;r)$ and $Q_h^\pi(s,a;r)$ to denote the value functions with respect to r as the reward function for disambiguation purposes. The optimal value functions and the optimal policy will be denoted by $V^*(s;r), Q^*(s,a;r), \pi_r^*$, respectively.

Non-Markov Policies While we focus on Markov policies in the above definition, some of our results apply to or require more general forms of policies. For example, our lower bounds apply to non-Markov policies that can depend on the history (e.g., $\mathcal{S}_1 \times \mathcal{A}_1 \times \mathbb{R} ... \times \mathcal{S}_{h-1} \times \mathcal{A}_{h-1} \times \mathbb{R} \times \mathcal{S}_h \to \mathcal{A}$ for deterministic policies); our algorithm for arbitrary policies deploys a mixture of deterministic Markov policies, which corresponds to choosing a deterministic policy from a given set at the initial state, and following that policy for the entire trajectory. This can be viewed as a non-Markov stochastic policy.

2.2 Linear MDP Setting

We mainly focus on the linear MDP (Jin et al., 2019) satisfying the following assumptions:

Assumption A (Linear MDP Assumptions)**.** An MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, H, P, r)$ is said to be a linear MDP with a feature map $\phi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$ if the following hold for any $h \in [H]$:

- There are d unknown signed measures $\mu_h = (\mu_h^{(1)}, \mu_h^{(2)}, ..., \mu_h^{(d)})$ over $\mathcal{S}$ such that for any $(s,a,s') \in \mathcal{S} \times \mathcal{A} \times \mathcal{S}, P_h(s'|s,a) = \langle \mu_h(s'), \phi(s,a) \rangle$.
- There exists an unknown vector $\theta_h \in \mathbb{R}^d$ such that for any $(s,a) \in \mathcal{S} \times \mathcal{A}$, $r_h(s,a) = \langle \phi(s,a), \theta_h \rangle$.

Similar to Jin et al. (2019) and Wang et al. (2020b), without loss of generality, we assume for all $(s,a) \in \mathcal{S} \times \mathcal{A}$ and $h \in [H]$, $\|\phi(s,a)\| \leqslant 1$, $\|\mu_h\| \leqslant \sqrt{d}$, and $\|\theta_h\| \leqslant \sqrt{d}$. In Section 3 we will refer to linear MDPs with stationary dynamics, which is a special case when $\mu_1 = \mu_2 = \ldots \mu_H$ and $\theta_1 = \theta_2 = \ldots = \theta_H$.

2.3 A Concrete Definition of DE-RL

In the following, we introduce our formulation for DE-RL in linear MDPs. For discussions of comparison to existing works, please refer to Section 1.1.

Definition 2.1 (Deployment Complexity in Linear MDPs)**.** We say that an algorithm has a deployment complexity K in linear MDPs if the following holds: given an arbitrary linear MDP under Assumption A, for arbitrary ε and $0 < \delta < 1$, the algorithm will return a policy π_K after K deployments and collecting at most N trajectories in each deployment, under the following constraints:

(a) With probability $1 - \delta$, π_K is ε-optimal, i.e. $J(\pi_K) \geqslant \max_\pi J(\pi) - \varepsilon$.

(b) The sample size N is polynominal, i.e. $N = \mathrm{poly}(d, H, \frac{1}{\varepsilon}, \log \frac{1}{\delta})$. Moreover, N should be fixed a priori and cannot change adaptively from deployment to deployment.

Under this definition, the goal of Deployment-Efficient RL is to design algorithms with provable guarantees of low deployment complexity.

Polynomial Size of N We emphasize that the restriction of polynomially large N is crucial to our formulation, and not including it can result in degenerate solutions. For example, if N is allowed to be exponentially large, we can finish exploration in 1 deployment in the arbitrary policy setting, by deploying a mixture of exponentially many policies that form an ε-net of the policy space. Alternatively, we can sample actions uniformly, and use importance sampling (Precup, 2000) to evaluate all of them in an off-policy manner. None of these solutions are practically feasible and are excluded by our restriction on N.

3 LOWER BOUND FOR DEPLOYMENT COMPLEXITY IN RL

In this section, we provide information-theoretic lower bounds of the deployment complexity in our DE-RL setting. We defer the lower bound construction and the proofs to Appendix B. As mentioned in Section 2, we consider non-Markov policies when we refer to deterministic and stochastic policies in this section, which strengthens our lower bounds as they apply to very general forms of policies.

We first study the algorithms which can only deploy deterministic policy at each deployment.

Theorem 3.1. *[Lower bound for deterministic policies, informal] For any $d \geqslant 4, H$ and any algorithm ψ that can only deploy a deterministic policy at each deployment, there exists a linear MDP M satisfying Assumption A, such that the deployment complexity of ψ in M is $K = \Omega(dH)$.*

The basic idea of our construction and the proof is that, intuitively, a linear MDP with dimension d and horizon length H has $\Omega(dH)$ "independent directions", while deterministic policies have limited exploration capacity and only reach $\Theta(1)$ direction in each deployment, which result in $\Omega(dH)$ deployments in the worst case.

In the next theorem, we will show that, even if the algorithm can use arbitrary exploration strategy (e.g. maximizing entropy, adding reward bonus), without additional assumptions, the number of deployments K still has to depend on H and may not be reduced to a constant when H is large.

Theorem 3.2. *[Lower bound for arbitrary policies, informal] For any $d \geqslant 4, H, N$ and any algorithm ψ which can deploy arbitrary policies, there exists a linear MDP M satisfying Assumption A, such that the deployment complexity of ψ in M is $K = \Omega(H/\lceil \log_d(NH) \rceil) = \widetilde{\Omega}(H)$.*

The origin of the difficulty can be illustrated by a recursive dilemma: in the worst case, if the agent does not have enough information at layer h, then it cannot identify a good policy to explore layer $h + \Omega(\log_d(NH))$ in 1 deployment, and so on and so forth. Given that we enforce N to be polynomial, the agent can only push the "information boundary" forward by $\Omega(\log_d(NH)) = \widetilde{\Omega}(1)$ layers per deployment. In many real-world applications, such difficulty can indeed exist. For example, in healthcare, the entire treatment is often divided into multiple stages. If the treatment in stage h is not effective, the patient may refuse to continue. This can result in insufficient samples for identifying a policy that performs well in stage $h + 1$.

Stationary vs. non-stationary dynamics Since we consider non-stationary dynamics in Assump. A, one may suspect that the H-dependence in the lower bound is mainly due to such non-stationarity. We show that this is not quite the case, and the H-dependence still exists for stationary dynamics. In fact, our lower bound for non-stationary dynamics directly imply one for stationary dynamics: given a finite horizon non-stationary MDP $\widetilde{M} = (\widetilde{\mathcal{S}}, \mathcal{A}, H, \widetilde{P}, \widetilde{r})$, we can construct a stationary MDP $M = (\mathcal{S}, \mathcal{A}, H, P, r)$ by expanding the state space to $\mathcal{S} = \widetilde{\mathcal{S}} \times [H]$ so that the new transition function P and reward function r are stationary across time steps. As a result, given arbitrary $d \geqslant 4$ and $H \geqslant 2$, we can construct a hard non-stationary MDP instance $\widetilde{M}$ with dimension $\widetilde{d} = \max\{4, d/H\}$ and horizon $\widetilde{h} = d/\widetilde{d} = \min\{H, d/4\}$, and convert it to a stationary MDP M with dimension d and horizon $h = \widetilde{h} = \min\{H, d/4\} \leqslant H$. If there exists an algorithm which can solve M in K deployments, then it can be used to solve $\widetilde{M}$ in no more than K deployments. Therefore, the lower bounds for stationary MDPs can be extended from Theorems 3.1 and 3.2, as shown in the following corollary:

Corollary 3.3 (Extension to Stationary MDPs)**.** *For stationary linear MDP with $d \geqslant 4$ and $H \geqslant 2$, suppose $N = \mathrm{poly}(d, H, \frac{1}{\varepsilon}, \log \frac{1}{\delta})$, the lower bound of deployment complexity would be $\Omega(d)$ for deterministic policy algorithms, and $\Omega(\frac{\min\{d/4, H\}}{\lceil \log_{\max\{d/H, 4\}} NH \rceil}) = \widetilde{\Omega}(\min\{d, H\})$ for algorithms which can deploy arbitrary policies.*

As we can see, the dependence on dimension and horizon will not be eliminated even if we make a stronger assumption that the MDP is stationary. The intuition is that, although the transition function is stationary, some states may not be reachable from the initial state distribution within a small number of times, so the stationary MDP can effectively have a "layered" structure. For example, in Atari games (Bellemare et al., 2013) (where many algorithms like DQN (Mnih et al., 2013) model the environments as infinite-horizon discounted MDPs) such as Breakout, the agent cannot observe

states where most of the bricks are knocked out at the initial stage of the trajectory. Therefore, the agent still can only push forward the "information frontier" a few steps per deployment. That said, it is possible reduce the deployment complexity lower bound in stationary MDPs by adding more assumptions, such as the initial state distribution providing good coverage over the entire state space, or all the states are reachable in the first few time steps. However, because these assumptions do not always hold and may overly trivialize the exploration problem, we will not consider them in our algorithm design. Besides, although our algorithms in the next section are designed for non-stationary MDPs, they can be extended to stationary MDPs by sharing covariance matrices, and we believe the analyses can also be extended to match the lower bound in Corollary 3.3.

4 Towards Optimal Deployment Efficiency

In this section we provide algorithms with deployment-efficiency guarantees that nearly match the lower bounds established in Section 3. Although our lower bound results in Section 3 consider non-Markov policies, our algorithms in this section only use Markov policies (or a mixture of Markov policies, in the arbitrary policy setting), which are simpler to implement and compute and are already near-optimal in deployment efficiency.

Inspiration from Lower Bounds: a Layer-by-Layer Exploration Strategy The linear dependence on H in the lower bounds implies a possibly deployment-efficient manner to explore, which we call a layer-by-layer strategy: conditioning on sufficient exploration in previous $h-1$ time steps, we can use $\mathrm{poly}(d)$ deployments to sufficiently explore the h-th time step, then we only need $H \cdot \mathrm{poly}(d)$ deployments to explore the entire MDP. If we can reduce the deployment cost in each layer from $\mathrm{poly}(d)$ to $\Theta(d)$ or even $\Theta(1)$, then we can achieve the optimal deployment efficiency. Besides, as another motivation, in Appendix C.4, we will briefly discuss the additional benefits of the layer-by-layer strategy, which will be useful especially in "Safe DE-RL". In Sections 4.1 and 4.2, we will introduce algorithms based on this idea and provide theoretical guarantees.

4.1 Deployment-Efficient RL with Deterministic Policies

Algorithm 1: Layer-by-Layer Batch Exploration Strategy for Linear MDPs Given Reward Function

Input: Failure probability $\delta > 0$, and target accuracy $\varepsilon > 0$, $\beta \leftarrow c_\beta \cdot dH\sqrt{\log(dH\delta^{-1}\varepsilon^{-1})}$ for some $c_\beta > 0$, total number of deployments K, batch size N,
$h_1 \leftarrow 1$ // h_k denotes the layer to explore in iteration k, for all $k \in [K]$
for $k = 1, 2, ..., K$ **do**
  $Q^k_{h_k+1}(\cdot,\cdot) \leftarrow 0$ and $V^k_{h_k+1}(\cdot) = 0$
  for $h = h_k, h_k - 1, ..., 1$ **do**
    $\Lambda^k_h \leftarrow I + \sum_{\tau=1}^{k}\sum_{n=1}^{N} \phi^{\tau n}_h (\phi^{\tau n}_h)^\top, \qquad u^k_h(\cdot,\cdot) \leftarrow \min\{\beta \cdot \sqrt{\phi(\cdot,\cdot)^\top (\Lambda^k_h)^{-1}\phi(\cdot,\cdot)}, H\}$
    $w^k_h \leftarrow (\Lambda^k_h)^{-1} \sum_{\tau=1}^{k-1}\sum_{n=1}^{N} \phi^{\tau n}_h \cdot V^k_{h+1}(s^{\tau n}_{h+1})$
    $Q^k_h(\cdot,\cdot) \leftarrow \min\{(w^k_h)^\top \phi(\cdot,\cdot) + r_h(\cdot,\cdot) + u^k_h(\cdot,\cdot), H\}$ and $V^k_h(\cdot) = \max_{a\in\mathcal{A}} Q^k_h(\cdot, a)$
    $\pi^k_h(\cdot) \leftarrow \arg\max_{a\in\mathcal{A}} Q^k_h(\cdot, a)$
  end
  Define $\pi^k = \pi^k_1 \circ \pi^k_2 ... \circ \pi^k_{h_k} \circ \mathrm{unif}_{[h_k+1:H]}$
  for $n = 1, ..., N$ **do**
    Receive initial state $s^{kn}_1 \sim d_1$
    for $h = 1, 2, ..., H$ **do** Take action $a^{kn}_h \leftarrow \pi^k_h(s^{kn}_h)$ and observe $s^{kn}_{h+1} \sim P_h(s^k_h, a^k_h)$;
  end
  Compute $\Delta_k \leftarrow \frac{2\beta}{N}\sum_{n=1}^{N}\sum_{h=1}^{h_k} \sqrt{\phi(s^{kn}_h, a^{kn}_h)^\top (\Lambda^k_h)^{-1}\phi(s^{kn}_h, a^{kn}_h)}$.
  if $\Delta_k \geqslant \frac{\varepsilon h_k}{2H}$ **then** $h_{k+1} \leftarrow h_k$;
  else if $h_k = H$ **then return** π^k ;
  else $h_{k+1} \leftarrow h_k + 1$;
end

In this sub-section, we focus on the setting where each deployed policy is deterministic. In Alg 1, we propose a provably deployment-efficient algorithm built on Least-Square Value Iteration with UCB (Jin et al., 2019)[4] and the "layer-by-layer" strategy. Briefly speaking, at deployment k, we focus on exploration in previous h_k layers, and compute $\pi_1^k, \pi_2^k, ..., \pi_{h_k}^k$ by running LSVI-UCB in an MDP truncated at step h_k. After that, we deploy π^k to collect N trajectories, and complete the trajectory after time step h_k with an arbitrary policy. (In the pseudocode we choose uniform, but the choice is inconsequential.) In line 19, we compute Δ_k with samples and use it to judge whether we should move on to the next layer till all H layers have been explored. The theoretical guarantee is listed below, and the missing proofs are deferred to Appendix C.

Theorem 4.1 (Deployment Complexity)**.** *For arbitrary $\varepsilon, \delta > 0$, and arbitrary $c_K \geqslant 2$, as long as $N \geqslant c\Big(c_K \frac{H^{4c_K+1}d^{3c_K}}{\varepsilon^{2c_K}} \log^{2c_K}(\frac{Hd}{\delta\varepsilon})\Big)^{\frac{1}{c_K-1}}$, where c is an absolute constant, by choosing*

$$K = c_K dH + 1. \tag{1}$$

Algorithm 1 will terminate at iteration $k \leqslant K$ and return us a policy π^k, and with probability $1-\delta$, $\mathbb{E}_{s_1\sim d_1}[V_1^*(s_1) - V_1^{\pi^k}(s_1)] \leqslant \varepsilon$.

As an interesting observation, Eq (1) reflects the trade-off between the magnitude of K and N when K is small. To see this, when we increase c_K and keep it at the constant level, K definitely increases while N will be lower because its dependence on $d, H, \varepsilon, \delta$ decreases. Moreover, the benefit of increasing c_K is only remarkable when c_K is small (e.g. we have $N = O(H^9d^6\varepsilon^{-4})$ if $c_K = 2$, while $N = O(H^5d^{3.6}\varepsilon^{-2.4})$ if $c_K = 6$), and even for moderately large c_K, the value of N quickly approaches the limit $\lim_{c_K\to\infty} N = c\frac{H^4d^3}{\varepsilon^2}\log^2(\frac{Hd}{\delta\varepsilon})$.

Another key step in proving the deployment efficiency of Algorithm 1 is Lemma 4.2 below. In Appendix C, we will discuss the possibility of applying this technique to LSVI-UCB (Jin et al., 2019) with large batch sizes, and the additional benefit of our layer-by-layer strategy.

Lemma 4.2. *[Finite Sample Elliptical Potential Lemma] Consider a sequence of matrices $\mathbf{A}_0, \mathbf{A}_N, ..., \mathbf{A}_{(K-1)N} \in \mathbb{R}^{d\times d}$ with $\mathbf{A}_0 = I_{d\times d}$ and $\mathbf{A}_{kN} = \mathbf{A}_{(k-1)N} + \Phi_{k-1}$, where $\Phi_{k-1} = \sum_{t=(k-1)N+1}^{kN} \phi_t\phi_t^\top$ and $\max_{t\leqslant KN}\|\phi_t\| \leqslant 1$. We define: $\mathcal{K}^+ := \Big\{k \in [K] \Big| Tr(\mathbf{A}_{(k-1)N}^{-1}\Phi_{k-1}) \geqslant N\varepsilon\Big\}$. For arbitrary $\varepsilon < 1$, and arbitrary $c_K \geqslant 2$, if $K = c_K dH + 1$, by choosing $N \geqslant c\Big(c_K\frac{Hd^{c_K}}{\varepsilon^{c_K}}\log^{c_K}(\frac{Hd}{\varepsilon})\Big)^{\frac{1}{c_K-1}}$, where c is an absolute constant independent with c_K, d, H, ε, we have $|\mathcal{K}^+| \leqslant c_K d < K/H$.*

Extension to Reward-free setting Based on the similar methodology, we can design algorithms for reward-free setting (Wang et al., 2020b) and obtain guarantees for deployment complexity based on Lemma 4.2. We defer the algorithms and proofs to Appendix D, and summarize the main result in Theorem D.4

4.2 Deployment-Efficient RL with Arbitrary Policies

From the discussion of lower bounds in Section 3, we know that in order to reduce the deployment complexity from $\Omega(dH)$ to $\widetilde{\Omega}(H)$, we have to utilize stochastic (and possibly non-Markov) policies and try to explore as many different directions as possible in each deployment (as opposed to 1 direction in Algorithm 1). The key challenge is to find a stochastic policy—before the deployment starts—which can sufficiently explore d independent directions.

In Algorithm 2, we overcome this difficulty by a new covariance matrix estimation method (Algorithm 6 in Appendix E). The basic idea is that, for arbitrary policy π [5], the covariance matrix $\Lambda_h^\pi := \mathbb{E}_\pi[\phi(s_h, a_h)\phi(s_h, a_h)^\top]$ can be estimated element-wise by running policy evaluation for

[4] In order to align with the algorithm in reward-free setting, slightly different from (Jin et al., 2019) but similar to (Wang et al., 2020b), we run linear regression on P_hV_h instead of Q_h.

[5] Here we mainly focus on evaluating deterministic policy or stochastic policy mixed from a finite number of determnistic policies, because for the other stochastic policies, exactly computing the expectation over policy distribution may be intractable.

Algorithm 2: Deployment-Efficient RL with Covariance Matrix Estimation

1 **Input**: Accuracy level ε; Iteration number $i_{\max}$; Resolution ε_0; Reward r; Bonus coefficient β.
2 **for** $h = 1, 2, ..., H$ **do**
3 Initialize $\pi_{h,1}$ with an arbitrary deterministic policy ; $\widetilde{\Sigma}_{h,1} = 2I$, $\Pi_h = \{\}$.
4 **for** $i = 1, 2, ..., i_{\max}$ **do**
5 $\widehat{\Lambda}_h^{\pi_{h,i}} \leftarrow \mathrm{EstimateCovMatrix}(h, D_{[1:h-1]}, \Sigma_{[1:h-1]}, \pi_{h,i})$ # Alg 6, Appx E
6 $\widetilde{\Sigma}_{h,i+1} = \widetilde{\Sigma}_{h,i} + \widehat{\Lambda}_h^{\pi_{h,i}}$
7 $V_{h,i+1},\ \bar{\pi}_{h,i+1} \leftarrow \mathrm{SolveOptQ}(h, D_{[1:h-1]}, \Sigma_{[1:h-1]}, \beta, \widetilde{\Sigma}_{h,i+1}, \varepsilon_0)$ # Alg 5, Appx E
8 **if** $V_{h,i+1} \leqslant 3\nu_{\min}^2/8$ **then** break ;
9 $\Pi_h = \Pi_h \bigcup \{\bar{\pi}_{h,i+1}\}$
10 **end**
11 $\Sigma_h = I$, $D_h = \{\}$, $\pi_{h,\mathrm{mix}} := \mathrm{unif}(\Pi_h)$
12 **for** $n = 1, 2, ..., N$ **do**
13 Sample trajectories with $\pi_{h,\mathrm{mix}}$
14 $\Sigma_h = \Sigma_h + \phi(s_{h,n}, a_{h,n})\phi(s_{h,n}, a_{h,n})^\top, \quad D_h = D_h \bigcup \{s_{h,n}, a_{h,n}, r_{h,n}, s_{h+1,n}\}$
15 **end**
16 **end**
17 **return** $\widehat{\pi}_r \leftarrow$ Alg 4$(H, \{D_1, ..., D_H\}, r)$

π with $\phi_i(s_h, a_h)\phi_j(s_h, a_h)$ as a reward function, where $i, j \in [d]$ and $\phi_i(\cdot, \cdot)$ denotes the i-th component of vector $\phi(\cdot, \cdot)$.

However, a new challenge emerging is that, because the transition is stochastic, in order to guarantee low evaluation error for all possible policies $\bar{\pi}_{h,i+1}$, we need an union bound over all policies to be evaluated, which is challenging if the policy class is infinite. To overcome this issue, we discretize the value functions in Algorithm 5 (see Appendix E) to allow for a union bound over the policy space: after computing the Q-function by LSVI-UCB, before converting it to a greedy policy, we first project it to an ε_0-net of the entire Q-function class. In this way, the number of policy candidates is finite and the projection error can be controlled as long as ε_0 is small enough.

Using the above techniques, in Lines 3-10, we repeatedly use Alg 6 to estimate the accumulative covariance matrix $\widetilde{\Sigma}_{h,i+1}$ and further eliminate uncertainty by calling Alg 5 to find a policy (approximately) maximizing uncertainty-based reward function $\widetilde{R} := \|\phi\|_{\widetilde{\Sigma}_{h,i+1}^{-1}}$. For each $h \in [H]$, inductively conditioning on sufficient exploration in previous $h-1$ layers, the errors of Alg 6 and Alg 5 will be small, and we will find a finite set of policies Π_h to cover all dimensions in layer h. (This is similar to the notion of "policy cover" in Du et al. (2019); Agarwal et al. (2020a).) Then, layer h can be explored sufficiently by deploying a uniform mixture of Π and choosing N large enough (Lines 11-15). Also note that the algorithm does not use the reward information, and is essentially a reward-free exploration algorithm. After exploring all H layers, we obtain a dataset $\{D_1, ..., D_H\}$ and can use Alg 4 for planning with any given reward function r satisfying Assump. A to obtain a near-optimal policy.

Deployment complexity guarantees We first introduce a quantity denoted as $\nu_{\min}$, which measures the reachability to each dimension in the linear MDP. In Appendix E.8, we will show that the $\nu_{\min}$ is no less than the "explorability" coefficient in Definition 2 of Zanette et al. (2020) and $\nu_{\min}^2$ is also lower bounded by the maximum of the smallest singular value of matrix $\mathbb{E}_\pi[\phi\phi^\top]$.

Definition 4.3 (Reachability Coefficient)**.**

$$\nu_h := \min_{\|\theta\|=1} \max_{\pi} \sqrt{\mathbb{E}_\pi[(\phi_h^\top \theta)^2]} \ ; \qquad \nu_{\min} = \min_{h \in [H]} \nu_h \ .$$

Now, we are ready to state the main theorem of this section, and defer the formal version and its proofs to Appendix E. Our algorithm is effectively running reward-free exploration and therefore our results hold for arbitrary linear reward functions.

Theorem 4.4. *[Informal] For arbitrary $0 < \varepsilon, \delta < 1$, with proper choices of $i_{\max}, \varepsilon_0, \beta$, we can choose $N = \mathrm{poly}(d, H, \frac{1}{\varepsilon}, \log\frac{1}{\delta}, \frac{1}{\nu_{\min}})$, such that, after $K = H$ deployments, with probability $1-\delta$,*

Algorithm 2 will collect a dataset $D = \{D_1, ..., D_H\}$, and if we run Alg 4 with D and arbitrary reward function satisfying Assump. A, we will obtain $\widehat{\pi}_r$ such that $V_1^{\widehat{\pi}_r}(s_1; r) \geqslant V_1^*(s_1; r) - \varepsilon$.

Proof Sketch Next, we briefly discuss the key steps of the proof. Since ε_0 can be chosen to be very small, we will ignore the bias induced by ε_0 when providing intuitions. Our proof is based on the induction condition below. We first assume it holds after $h-1$ deployments (which is true when $h = 1$), and then we try to prove at the h-th deployment we can explore layer h well enough so that the condition holds for h.

Condition 4.5. [Induction Condition] Suppose after $h-1$ deployments, we have the following induction condition for some $\xi < 1/d$, which will be determined later:

$$\max_\pi \mathbb{E}_\pi[\textstyle\sum_{\widetilde{h}=1}^{h-1} \sqrt{\phi(s_{\widetilde{h}}, a_{\widetilde{h}})^\top \Sigma_{\widetilde{h}}^{-1} \phi(s_{\widetilde{h}}, a_{\widetilde{h}})}] \leqslant \tfrac{h-1}{H}\xi. \tag{2}$$

The l.h.s. of Eq.(2) measures the uncertainty in previous $h-1$ layers after exploration. As a result, with high probability, the following estimations will be accurate:

$$\|\widehat{\Lambda}^{\pi_{h,i}} - \mathbb{E}_{\pi_{h,i}}[\phi(s_h, a_h)\phi(s_h, a_h)^\top]\|_{\infty,\infty} \leqslant O(\xi), \tag{3}$$

where $\|\cdot\|_{\infty,\infty}$ denotes the entry-wise maximum norm. This directly implies that:

$$\|\widetilde{\Sigma}_{h,i+1} - \Sigma_{h,i+1}]\|_{\infty,\infty} \leqslant i \cdot O(\xi).$$

where $\Sigma_{h,i+1} := 2I + \sum_{i'=1}^{i} \mathbb{E}_{\pi_{h,i'}}[\phi(s_h, a_h)\phi(s_h, a_h)^\top]$ is the target value for $\widetilde{\Sigma}_{h,i+1}$ to approximate. Besides, recall that in Algorithm 5, we use $\sqrt{\phi^\top \widetilde{\Sigma}_{h,i+1}^{-1} \phi}$ as the reward function, and the induction condition also implies that:

$$|V_{h,i+1} - \max_\pi \mathbb{E}_\pi[\|\phi(s_h, a_h)\|_{\widetilde{\Sigma}_{h,i+1}^{-1}}]| \leqslant O(\xi).$$

As a result, if ξ and the resolution ε_0 are small enough, $\bar{\pi}_{h,i+1}$ would gradually reduce the uncertainty and $V_{h,i+1}$ (also $\max_\pi \mathbb{E}_\pi[\|\phi(s_h, a_h)\|_{\widetilde{\Sigma}_{h,i+1}^{-1}}]$) will decrease. However, the bias is at the level $O(\xi)$, and therefore, no matter how small ξ is, as long as $\xi > 0$, it is still possible that the policies in Π_h do not cover all directions if some directions are very difficult to reach, and the error due to such a bias will be at the same level of the required accuracy in induction condition, i.e. $O(\xi)$. This is exactly where the "reachability coefficient" $\nu_{\min}$ definition helps. The introduction of $\nu_{\min}$ provides a threshold, and as long as ξ is small enough so that the bias is lower than the threshold, each dimension will be reached with substantial probability when the breaking criterion in Line 9 is satisfied. As a result, by deploying $\mathrm{unif}(\Pi_h)$ and collecting a sufficiently large dataset, the induction condition will hold till layer H. Finally, combining the guarantee of Alg 4, we complete the proof.

5 Conclusion and Future Work

In this paper, we propose a concrete theoretical formulation for DE-RL to fill the gap between existing RL literatures and real-world applications with deployment constraints. Based on our framework, we establish lower bounds for deployment complexity in linear MDPs, and provide novel algorithms and techniques to achieve optimal deployment efficiency. Besides, our formulation is flexible and can serve as building blocks for other practically relevant settings related to DE-RL. We conclude the paper with two such examples, defer a more detailed discussion to Appendix F, and leave the investigation to future work.

Sample-Efficient DE-RL In our basic formulation in Definition 2.1, we focus on minimizing the deployment complexity K and put very mild constraints on the per-deployment sample complexity N. In practice, however, the latter is also an important consideration, and we may face additional constraints on how large N can be, as they can be upper bounded by e.g. the number of customers or patients our system is serving.

Safe DE-RL In real-world applications, safety is also an important criterion. The definition for safety criterion in Safe DE-RL is still an open problem, but we believe it is an interesting setting since it implies a trade-off between exploration and exploitation in deployment-efficient setting.

ACKNOWLEDGEMENTS

JH's research activities on this work were completed by December 2021 during his internship at MSRA. NJ acknowledges funding support from ARL Cooperative Agreement W911NF-17-2-0196, NSF IIS-2112471, and Adobe Data Science Research Award.

REFERENCES

Yasin Abbasi-yadkori, Dávid Pál, and Csaba Szepesvári. Improved algorithms for linear stochastic bandits. In *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011.

M Mehdi Afsar, Trafford Crump, and Behrouz Far. Reinforcement learning based recommender systems: A survey. *arXiv preprint arXiv:2101.06286*, 2021.

Alekh Agarwal, Mikael Henaff, Sham Kakade, and Wen Sun. Pc-pg: Policy cover directed exploration for provable policy gradient learning. *arXiv preprint arXiv:2007.08459*, 2020a.

Alekh Agarwal, Sham Kakade, Akshay Krishnamurthy, and Wen Sun. Flambe: Structural complexity and representation learning of low rank mdps, 2020b.

Priyank Agrawal, Jinglin Chen, and Nan Jiang. Improved worst-case regret bounds for randomized least-squares value iteration. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 6566–6573, 2021.

Shipra Agrawal and Randy Jia. Posterior sampling for reinforcement learning: worst-case regret bounds. In *Advances in Neural Information Processing Systems*, pp. 1184–1194, 2017.

András Antos, Csaba Szepesvári, and Rémi Munos. Learning near-optimal policies with bellman-residual minimization based fitted policy iteration and a single sample path. *Machine Learning*, 71(1):89–129, 2008.

M. G. Azar, Ian Osband, and R. Munos. Minimax regret bounds for reinforcement learning. In *ICML*, 2017.

Yu Bai, Tengyang Xie, Nan Jiang, and Yu-Xiang Wang. Provably efficient q-learning with low switching cost, 2020.

Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47: 253–279, 2013.

Marc G. Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation, 2016.

Abdellah Bennane et al. Adaptive educational software by applying reinforcement learning. *Informatics in Education-An International Journal*, 12(1):13–27, 2013.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation, 2018.

Andres Campero, Roberta Raileanu, Heinrich Küttler, Joshua B Tenenbaum, Tim Rocktäschel, and Edward Grefenstette. Learning with amigo: Adversarially motivated intrinsic goals. *arXiv preprint arXiv:2006.12122*, 2020.

Alexandra Carpentier, Claire Vernade, and Yasin Abbasi-Yadkori. The elliptical potential lemma revisited, 2020.

Jinglin Chen and Nan Jiang. Information-theoretic considerations in batch reinforcement learning. In *International Conference on Machine Learning*, pp. 1042–1051. PMLR, 2019.

Christoph Dann, Nan Jiang, Akshay Krishnamurthy, Alekh Agarwal, John Langford, and Robert E Schapire. On oracle-efficient pac rl with rich observations. *Advances in neural information processing systems*, 31, 2018.

Simon Du, Akshay Krishnamurthy, Nan Jiang, Alekh Agarwal, Miroslav Dudik, and John Langford. Provably efficient rl with rich observations via latent state decoding. In *International Conference on Machine Learning*, pp. 1665–1674. PMLR, 2019.

Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune. Go-explore: a new approach for hard-exploration problems. *arXiv preprint arXiv:1901.10995*, 2019.

Hossein Esfandiari, Amin Karbasi, Abbas Mehrabian, and Vahab Mirrokni. Regret bounds for batched bandits, 2020.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning, 2021.

Minbo Gao, Tianle Xie, Simon S. Du, and Lin F. Yang. A provably efficient algorithm for linear markov decision process with low switching cost, 2021.

Quanquan Gu, Amin Karbasi, Khashayar Khosravi, Vahab Mirrokni, and Dongruo Zhou. Batched neural bandits, 2021.

Zhaohan Guo and Emma Brunskill. Concurrent pac rl. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

Yanjun Han, Zhengqing Zhou, Zhengyuan Zhou, Jose Blanchet, Peter W. Glynn, and Yinyu Ye. Sequential batch learning in finite-action linear contextual bandits, 2020.

Elad Hazan, Sham M. Kakade, Karan Singh, and Abby Van Soest. Provably efficient maximum entropy exploration, 2019.

Nan Jiang and Alekh Agarwal. Open problem: The dependence of sample complexity lower bounds on planning horizon. In *Conference On Learning Theory*, pp. 3395–3398. PMLR, 2018.

Nan Jiang and Lihong Li. Doubly robust off-policy value evaluation for reinforcement learning. In *International Conference on Machine Learning*, pp. 652–661. PMLR, 2016.

Nan Jiang, Akshay Krishnamurthy, Alekh Agarwal, John Langford, and Robert E Schapire. Contextual decision processes with low bellman rank are pac-learnable. In *International Conference on Machine Learning*, pp. 1704–1713. PMLR, 2017a.

Nan Jiang, Akshay Krishnamurthy, Alekh Agarwal, John Langford, and Robert E. Schapire. Contextual decision processes with low Bellman rank are PAC-learnable. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pp. 1704–1713, 2017b.

Chi Jin, Zeyuan Allen-Zhu, Sebastien Bubeck, and Michael I. Jordan. Is q-learning provably efficient?, 2018.

Chi Jin, Zhuoran Yang, Zhaoran Wang, and Michael I. Jordan. Provably efficient reinforcement learning with linear function approximation, 2019.

Chi Jin, Akshay Krishnamurthy, Max Simchowitz, and Tiancheng Yu. Reward-free exploration for reinforcement learning, 2020.

Chi Jin, Qinghua Liu, and Sobhan Miryoosefi. Bellman eluder dimension: New rich classes of rl problems, and sample-efficient algorithms. *Advances in Neural Information Processing Systems*, 34, 2021a.

Ying Jin, Zhuoran Yang, and Zhaoran Wang. Is pessimism provably efficient for offline rl? In *International Conference on Machine Learning*, pp. 5084–5096. PMLR, 2021b.

Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

Dingwen Kong, R. Salakhutdinov, Ruosong Wang, and Lin F. Yang. Online sub-sampling for reinforcement learning with general function approximation. *ArXiv*, abs/2106.07203, 2021.

Akshay Krishnamurthy, Alekh Agarwal, and John Langford. Pac reinforcement learning with rich observations. *Advances in Neural Information Processing Systems*, 29:1840–1848, 2016.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *arXiv preprint arXiv:2006.04779*, 2020.

Romain Laroche, Paul Trichelair, and Remi Tachet Des Combes. Safe policy improvement with baseline bootstrapping. In *International Conference on Machine Learning*, pp. 3652–3661. PMLR, 2019.

Jongmin Lee, Wonseok Jeon, Byung-Jun Lee, Joelle Pineau, and Kee-Eung Kim. Optidice: Offline policy optimization via stationary distribution correction estimation. *arXiv preprint arXiv:2106.10783*, 2021.

Qiang Liu, Lihong Li, Ziyang Tang, and Dengyong Zhou. Breaking the curse of horizon: Infinite-horizon off-policy estimation. *arXiv preprint arXiv:1810.12429*, 2018.

Yao Liu, Adith Swaminathan, Alekh Agarwal, and Emma Brunskill. Provably good batch off-policy reinforcement learning without great exploration. *Advances in Neural Information Processing Systems*, 33:1264–1274, 2020.

Tatsuya Matsushima, Hiroki Furuta, Yutaka Matsuo, Ofir Nachum, and Shixiang Gu. Deployment-efficient reinforcement learning via model-based offline optimization. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=3hGNqpI4WS`.

Dipendra Misra, Mikael Henaff, Akshay Krishnamurthy, and John Langford. Kinematic state abstraction and provably efficient rich-observation reinforcement learning. In *International conference on machine learning*, pp. 6961–6971. PMLR, 2020.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Aditya Modi, Jinglin Chen, Akshay Krishnamurthy, Nan Jiang, and Alekh Agarwal. Model-free representation learning and exploration in low-rank mdps. *arXiv preprint arXiv:2102.07035*, 2021.

Ted Moskovitz, Jack Parker-Holder, Aldo Pacchiano, Michael Arbel, and Michael I. Jordan. Tactical optimism and pessimism for deep reinforcement learning, 2021.

Rémi Munos and Csaba Szepesvári. Finite-time bounds for fitted value iteration. *Journal of Machine Learning Research*, 9(5), 2008.

Ofir Nachum, Bo Dai, Ilya Kostrikov, Yinlam Chow, Lihong Li, and Dale Schuurmans. Algaedice: Policy gradient from arbitrary experience. *arXiv preprint arXiv:1912.02074*, 2019.

Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. Awac: Accelerating online reinforcement learning with offline datasets, 2021.

Karl Pertsch, Youngwoon Lee, and Joseph J Lim. Accelerating reinforcement learning with learned skill priors. *arXiv preprint arXiv:2010.11944*, 2020.

Doina Precup. Eligibility traces for off-policy policy evaluation. *Computer Science Department Faculty Publication Series*, pp. 80, 2000.

Yufei Ruan, Jiaqi Yang, and Yuan Zhou. Linear bandits with limited adaptivity and learning distributional optimal design, 2021.

Daniel Russo and Benjamin Van Roy. Eluder dimension and the sample complexity of optimistic exploration. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

Joel A. Tropp. An introduction to matrix concentration inequalities, 2015.

Masatoshi Uehara, Jiawei Huang, and Nan Jiang. Minimax weight and q-function learning for off-policy evaluation. In *International Conference on Machine Learning*, pp. 9659–9668. PMLR, 2020.

Ruosong Wang, Simon S. Du, Lin F. Yang, and Sham M. Kakade. Is long horizon reinforcement learning more difficult than short horizon reinforcement learning?, 2020a.

Ruosong Wang, Simon S. Du, Lin F. Yang, and Ruslan Salakhutdinov. On reward-free reinforcement learning with linear function approximation, 2020b.

Ruosong Wang, Ruslan Salakhutdinov, and Lin F Yang. Reinforcement learning with general value function approximation: Provably efficient approach via bounded eluder dimension. *arXiv preprint arXiv:2005.10804*, 2020c.

Tengyang Xie, Ching-An Cheng, Nan Jiang, Paul Mineiro, and Alekh Agarwal. Bellman-consistent pessimism for offline reinforcement learning. *arXiv preprint arXiv:2106.06926*, 2021a.

Tengyang Xie, Nan Jiang, Huan Wang, Caiming Xiong, and Yu Bai. Policy finetuning: Bridging sample-efficient offline and online reinforcement learning, 2021b.

Mengjiao Yang, Ofir Nachum, Bo Dai, Lihong Li, and Dale Schuurmans. Off-policy evaluation via the regularized lagrangian. *arXiv preprint arXiv:2007.03438*, 2020.

Chao Yu, Jiming Liu, and Shamim Nemati. Reinforcement learning in healthcare: A survey. *arXiv preprint arXiv:1908.08796*, 2019.

Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. *arXiv preprint arXiv:2005.13239*, 2020.

Andrea Zanette and Emma Brunskill. Tighter problem-dependent regret bounds in reinforcement learning without domain knowledge using value function bounds. In *International Conference on Machine Learning*, pp. 7304–7312. PMLR, 2019.

Andrea Zanette, Alessandro Lazaric, Mykel J Kochenderfer, and Emma Brunskill. Provably efficient reward-agnostic navigation with linear value iteration. *arXiv preprint arXiv:2008.07737*, 2020.

Tianjun Zhang, Paria Rashidinejad, Jiantao Jiao, Yuandong Tian, Joseph Gonzalez, and Stuart Russell. Made: Exploration via maximizing deviation from explored regions. *arXiv preprint arXiv:2106.10268*, 2021.

A EXTENDED RELATED WORK

Online RL Online RL is a paradigm focusing on the challenge of strategic exploration. On the theoretical side, based on the "Optimism in Face of Uncertainty"(OFU) principle or posterior sampling techniques, many provable algorithms have been developed for tabular MDPs (Jin et al., 2018; Azar et al., 2017; Zanette & Brunskill, 2019; Agrawal & Jia, 2017; Agrawal et al., 2021), linear MDPs (Jin et al., 2019; Agarwal et al., 2020a), general function approximation (Wang et al., 2020c; Russo & Van Roy, 2013), or MDPs with structural assumptions (Jiang et al., 2017b; Du et al., 2019). Moreover, there is another stream of work studying how to guide exploration by utilizing state occupancy (Hazan et al., 2019; Zhang et al., 2021). Beyond the learning in MDPs with pre-specified reward function, recently, Jin et al. (2020); Wang et al. (2020b); Zanette et al. (2020) provide algorithms for exploration in the scenarios where multiple reward functions are of interest. On the practical side, there are empirical algorithms such as intrinsically-motivated exploration (Bellemare et al., 2016; Campero et al., 2020), exploration with hand-crafted reward bonus (RND) (Burda et al., 2018), and other more sophisticated strategies (Ecoffet et al., 2019). However, all of these exploration methods do not take deployment efficiency into consideration, and will fail to sufficiently explore the MDP and learn near-optimal policies in DE-RL setting where the number of deployments is very limited.

Offline RL Different from the online setting, where the agents are encouraged to explore rarely visited states to identify the optimal policy, the pure offline RL setting serves as a framework for utilizing historical data to learn a good policy without further interacting with the environment. Therefore, the core problem of offline RL is the performance guarantee of the deployed policy, which motivated multiple importance-sampling based off-policy policy evaluation and optimization methods (Jiang & Li, 2016; Liu et al., 2018; Uehara et al., 2020; Yang et al., 2020; Nachum et al., 2019; Lee et al., 2021), and the "Pessimism in Face of Uncertainty" framework (Liu et al., 2020; Kumar et al., 2020; Fujimoto & Gu, 2021; Yu et al., 2020; Jin et al., 2021b; Xie et al., 2021a) in contrast with OFU in online exploration setting. However, as suggested in Matsushima et al. (2021), pure offline RL can be regarded as constraining the total number of deployments to be 1.

Bridging Online and Offline RL; Trade-off between Pessimism and Optimism As pointed out by Xie et al. (2021b); Matsushima et al. (2021), there is a clear gap between existing online and offline RL settings, and some efforts have been made towards bridging them. For example, Nair et al. (2021); Pertsch et al. (2020); Bai et al. (2020) studied how to leverage pre-collected offline datasets to learn a good prior to accelerate the online learning process. Moskovitz et al. (2021) proposed a learning framework which can switch between optimism and pessimism by modeling the selection as a bandit problem. None of these works give provable guarantees in our deployment-efficiency setting.

Conversion from Linear Regret in Gao et al. (2021) to Sample Complexity Gao et al. (2021) proposed an algorithm with the following guarantee: after interacting with the environments for $\widetilde{K}$ times (we use $\widetilde{K}$ to distinguish with K in our setting), there exists a constant c such that the algorithm's regret is

$$\sum_{k=1}^{\widetilde{K}} V^*(s_1) - V^{\pi_k}(s_1) = c \cdot \sqrt{d^3 H^4 \widetilde{K}} \cdot \iota = c \cdot \sqrt{d^3 H^3 T} \cdot \iota$$

where we denote $T = \widetilde{K} H$, and use ι to refer to the log terms. Besides, in $\pi_1, ..., \pi_K$ there are only $O(dH \log \widetilde{K})$ policy switching.

As discussed in Section 3.1 by Jin et al. (2018), such a result can be convert to a PAC guarantee that, by uniformly randomly select a policy π from $\pi_1, ..., \pi_K$, with probability at least 2/3, we should have:

$$V^*(s_1) - V^{\pi}(s_1) = \widetilde{O}(\sqrt{\frac{d^3 H^5}{T}}) = \widetilde{O}(\sqrt{\frac{d^3 H^4}{\widetilde{K}}})$$

In order to make sure the upper bound in the r.h.s. will be ε, we need:

$$\widetilde{K} = \frac{d^3 H^4}{\varepsilon^2}$$

and the required policy switching would be:

$$O(dH \log \widetilde{K}) = O(dH \log \frac{dH}{\varepsilon})$$

In contrast with our results in Section 4.1, there is an additional logarithmic dependence on d, H and ε. Moreover, since their algorithm only deploys deterministic policies, their deployment complexity has to depend on d, which is much higher than our stochastic policy algorithms in Section 4.2 when d is large.

Methods	R-F?	Deployed Policy	# Trajectories	Deployment Complexity
LSVI-UCB (Jin et al., 2019)	×	Deterministic	$\widetilde{O}(\frac{d^3H^4}{\varepsilon^2})$	
Reward-free LSVI-UCB (Wang et al., 2020b)	✓	Deterministic	$\widetilde{O}(\frac{d^3H^6}{\varepsilon^2})$	
FRANCIS (Zanette et al., 2020)	✓	Deterministic	$\widetilde{O}(\frac{d^3H^5}{\varepsilon^2})$	
Gao et al. (2021)	×	Deterministic	$\widetilde{O}(\frac{d^3H^4}{\varepsilon^2})$	$O(dH \log \frac{dH}{\varepsilon})$
Q-type OLIVE (Jiang et al., 2017a) (Jin et al., 2021a)	×	Deterministic	$\widetilde{O}(\frac{d^3H^6}{\varepsilon^2})$	$O(dH \log \frac{1}{\varepsilon})$
Simplified MOFFLE (Modi et al., 2021)	✓	Stochastic	$\widetilde{O}(\frac{d^8H^7\lvert\mathcal{A}\rvert^{13}}{\min(\varepsilon^2\eta_{\min},\eta_{\min}^5)})$	$\widetilde{O}(\frac{Hd^3\lvert\mathcal{A}\rvert^4}{\eta_{\min}^2})$
Alg. 1 [Ours]	×	Deterministic	$\widetilde{O}(\frac{d^4H^5}{\varepsilon^2})$	$O(dH)$
Alg. 3 + 4[Ours]	✓	Deterministic	$\widetilde{O}(\frac{d^4H^7}{\varepsilon^2})$	$O(dH)$
Alg. 2[Ours]	✓	Stochastic	$\widetilde{O}(\frac{d^7H^3}{\varepsilon^2\nu_{\min}^{14}})$	H

Table 1: Comparison between our algorithms and online RL methods without considering deployment costraints in our setting defined in Def. 2.1, where R-F is the short note for Reward-Free. The total number of trajectories cost by our methods is computed by $K \cdot N$. We omit log terms in $\widetilde{O}$. For algorithm (Jin et al., 2019), we report the sample complexity after the conversion from regret. For our deterministic policy algorithms, we report the asymptotic results when $c_K \to +\infty$, which can be achieved approximately when c_K is a large constant (e.g. $c_K = 100$).

Investigation on Trade-off between Sample Complexity and Deployment Complexity In Table 1, we compare our algorithms and previous online RL works which did not consider deployment efficiency to shed light on the trade-off between sample and deployment complexities. Besides algorithms that are specialized to linear MDPs, we also include results such as Zanette et al. (2020), which studied a more general linear approximation setting and can be adapted to our setting. As stated in Def. 2 of Zanette et al. (2020), they also rely on some reachability assumption. To avoid ambiguity, we use $\widetilde{\nu}_{\min}$ to refer to their reachability coefficient (as discussed in Appx E.8, $\widetilde{\nu}_{\min}$ is no larger than and can be much smaller than our $\nu_{\min}$). Because they also assume that $\varepsilon \leqslant \widetilde{O}(\widetilde{\nu}_{\min}/\sqrt{d})$ (see Thm 4.1 in their paper), their results have an implicit dependence on $\widetilde{\nu}_{\min}^{-2}$. In addition, by using the class of linear functions w.r.t. ϕ, Q-type OLIVE (Jiang et al., 2017a; Jin et al., 2021a) has $\widetilde{O}(\frac{d^3H^6}{\varepsilon^2})$ sample complexity and $O(dH \log(1/\varepsilon))$ deployment complexity. Its deployment complexity is close to our deterministic algorithm, but with additional dependence on ε. We also want to highlight that OLIVE is known to be computationally intractable (Dann et al., 2018), while our algorithms are computationally efficient. With the given feature ϕ in linear MDPs and additional reachability assumption (not comparable to us), we can use a simplified version of MOFFLE (Modi et al., 2021) by skipping their LearnRep subroutine. Though this version of MOFFLE is computationally efficient and its deployment complexity does not depend on ε, it has much worse sample complexity ($\eta_{\min}$ is their reachability coefficient) and deployment complexity. On the other hand, PCID (Du et al., 2019) and HOMER (Misra et al., 2020) achieve H deployment

complexity in block MDPs. However, block MDPs are more restricted than linear MDPs and these algorithms have worse sample and computational complexities.

It is worth to note that all our algorithms achieve the optimal dependence on ε (i.e., ε^{-2}) in sample complexity. For algorithms that deploy deterministic policies, we can see that our algorithm has higher dependence on d and H in the sample complexity in both reward-known and reward-free setting, while our deployment complexity is much lower. Our stochastic policy algorithm (last row) is naturally a reward-free algorithm. Comparing with Wang et al. (2020b) and Zanette et al. (2020), our sample complexity has higher dependence on d and the reachability coefficient $\nu_{\min}$, while our algorithm achieves the optimal deployment complexity and better dependence on H.

B On the Lower Bound of Deployment-Efficient Reinforcement Learning

B.1 A Hard MDP Instance Template

In this sub-section, we first introduce a hard MDP template that is used in further proofs. As shown in Figure 1, we construct a tabular MDP (which is a special case of linear MDP) where the horizon length is $H+1$ and in each layer except the first one, there are $d+2$ states and $2d+1$ different state action pairs. The initial state is fixed as s_0 and there are $d+2$ different actions. It is easy to see that we can represent the MDP by linear features with at most $2d+1$ dimensions, and construct reward and transition function satisfying Assumption A. As a result, it is a linear MDP with dimension $2d+1$ and horizon length $H+1$. Since there is only a constant-level blow up of dimension, the dimension of these MDPs is still $\Theta(d)$, and we will directly use d instead of $\Theta(d)$ in the rest of the proof. The states in each layer $h \geqslant 1$ can be divided into three groups and we introduce them one-by-one in the following.

Group 1: Absorbing States (Green Color) The first group $G_h^1 = \{u_h^1, u_h^2\}$ consists of two absorbing states u_h^1 and u_h^2, which can only take one action at each state $\bar{a}_h^1$ and $\bar{a}_h^2$ and transit to u_{h+1}^1 and u_{h+1}^2 with probability 1, respectively. The reward function is defined as $r_h(u_h^1, \bar{a}_h^1) = r_h(u_h^2, \bar{a}_h^2) = 0.5$ for all $h \leqslant H-2$ and $r_H(u_H^1, \bar{a}_H^1) = 0.0$, $r_H(u_H^2, \bar{a}_H^2) = 1.0$.

Group 2: Core States (Red Color) The second group $G_h^2 = \{s_h^*\}$ only contains one state, which we call it core state and denote it as s_h^*. For example, in Figure 1, we have $s_1^* = s_1^d, s_2^* = s_2^1$ and $s_3^* = s_3^2$. In the "core state", the agent can take d actions $\widetilde{a}_h^1, \widetilde{a}_h^2, \ldots, \widetilde{a}_h^d$ and transit deterministically to $s_{h+1}^1, s_{h+1}^2, \ldots, s_{h+1}^d$. Besides, the reward function is $r_h(s_h^*, \widetilde{a}_h^i) = 0.5$ for all $i \in [d]$.

Group 3: Normal States (Blue Color) The third group $G_h^3 = \{s_h^i | i \in [d], s_h^i \neq s_h^*\}$ is what we call "normal states", and each state $s_h^i \in G_h^3$ can only take one action $\widetilde{a}_h^i$ and will transit randomly to one of the absorbing states in the next layer, i.e. G_{h+1}^1. Besides, the reward function is $r_h(s_h^i, \widetilde{a}_h^i) = 0.5$ for arbitrary $s_h^i \in G_h^3$, and the transition function is $P(u_{h+1}^1 | s_h^i, \widetilde{a}_h^i) = P(u_{h+1}^1 | s_h^i, \widetilde{a}_h^i) = 0.5$, except for a state action pair $s^\# := s_{h^\#}^{i^\#}, a^\# := a_{h^\#}^{i^\#}$ at layer $h^\# \in [H-1]$ with index $i^\#$, such that $s^\# \notin G_{h^\#}^2$ and $P(u_{h+1}^1 | s_{h^\#}^{i^\#}, a_{h^\#}^{i^\#}) = 0.5 - \varepsilon$ and $P(u_{h+1}^2 | s_{h^\#}^{i^\#}, a_{h^\#}^{i^\#}) = 0.5 + \varepsilon$. In the following, we will call $s^\#, a^\#$ the "optimal state" and "optimal action" in this MDP. Note that the "optimal state" can not be the core state.

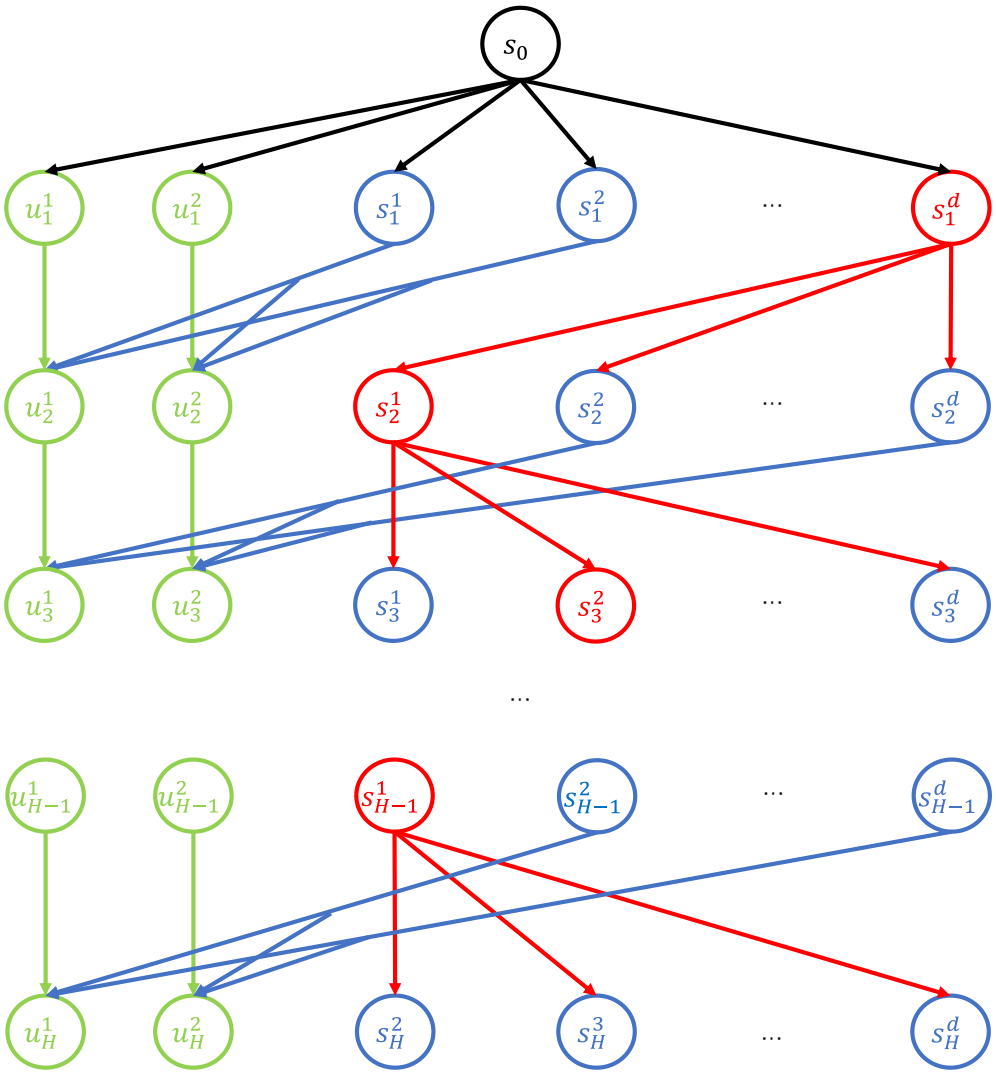
We will use $M(h^\#, i^\#, I_{core} = \{i_1, i_2, ..., i_H\})$ with $i_{h^\#} \neq i^\#$ to denote the MDP whose optimal state is at layer $h^\#$ and indexed by $i^\#$, and the core states in each layer are $s_h^{i_1}, s_h^{i_2}, ..., s_H^{i_H}$. As we can see, the only optimal policy should be the one which can generate the following sequence of states before transiting to absorb states at layer $h^\# + 1$:

$$s_0, s_1^{i_1}, s_2^{i_2}, ..., s_{h^\#-1}^{i_{h^\#-1}}, s_{h^\#}^{i^\#}$$

and the optimal value function would be $\frac{1}{2}H + \varepsilon$. In order to achieve ε-optimal policy, the algorithm should identify $s_{h^\#}^{i^\#}$, which is the main origin of the difficulty in exploration.

Remark B.1 (Markov v.s. Non-Markov Policies)**.** *As we can see, the core states in each layer are the only states with #actions > 1, and for each core state, there exists and only exists one*

Figure 1: Lower Bound Instance Template. The states in each layer can be divided into three groups. **Group 1**: absorbing states, marked with green; **Group 2**: core state, marked with red; **Group 3**: normal states, marked with blue.



deterministic path (a sequence of states, actions and rewards) from initial state to it, which implies that for arbitrary non-Markov policy, there exists an equivalent Markov policy. Therefore, in the rest of the proofs in this section, we only focus on Markov policies.

B.2 Lower bound for algorithms which can deploy deterministic policies only

In the following, we will state the formal version of the lower bound theorem for deterministic policy setting and its proof. The intuition of the proof is that we can construct a hard instance, which can be regarded as a $\Omega(dH)$ multi-arm bandit problem, and we will show that in expectation the algorithm need to "pull $\Omega(dH)$ arms" before identifying the optimal one.

Theorem B.2 (Lower bound for number of deployments in deterministic policy setting)**.** *For the linear MDP problem with dimension d and horizon H, given arbitrary algorithm ψ (ψ can be deterministic or stochastic), which can only deploy a deterministic policy but can collect arbitrary*

number of samples in each deployment, there exists a MDP problem where the optimal deterministic policy π^* is ε better than all the other deterministic policies, but the estimate policy $\widehat{\pi}$ (which is also a deterministic policy) of the best policy output by ψ after K deployments must have $P(\pi^* \neq \widehat{\pi}) \geqslant 1/10$ unless the number of deployments $K > (d-1)(H-1)/2 = \Omega(dH)$.

Proof. First of all, we introduce how we construct hard instances.

Construction of Hard Instances We consider a set of MDPs $\bar{\mathcal{M}}$, where for each MDP in that set, the core states (red color) in each layer are fixed to be $s_1^1, s_2^1, ..., s_{H-1}^1$ and the only optimal states which has different probability to transit to absorbing states are randomly selected from $(d-1)(H-1)$ normal states (blue color). Easy to see that, $|\bar{\mathcal{M}}| = (d-1)(H-1)$.

Because of the different position of optimal states, the optimal policies for each MDP in $\bar{\mathcal{M}}$ (i.e. the policy which can transit from s_0 to optimal state) is different. We will use $\pi_1, \pi_2, ..., \pi_{(d-1)(H-1)}$ to refer to those different policies and use M_{π_i} with $1 \leqslant i \leqslant (d-1)(H-1)$ to denote the MDP in $\bar{\mathcal{M}}$ where π_i is the optimal policy. For convenience, we will use M_0 to denote the MDP where all the normal states have equal probability to transit to different absorbing states, i.e., all states are optimal states. Based on the introduction above, we define $\mathcal{M} := \bar{\mathcal{M}} \bigcup \{M_0\}$ and use the MDPs in $\mathcal{M}$ as hard instances.

Lower Bound for Average Failure Probability Next, we try to lower bound the average failure probability, which works as a lower bound for the maximal failure probability among MDPs in $\mathcal{M}$. Since any randomized algorithm is just a distribution over deterministic ones, and it therefore suffices to only consider deterministic algorithms ψ (Krishnamurthy et al., 2016).

Given an arbitrary algorithm ψ and $k \in [K]$, we use $\psi(k)$ to denote the policy taken by ψ at the k-th deployment (which is a random variable). Besides, we denote $\psi(K+1)$ as the output policy.

For arbitrary $k \in [K]$, we use $P_{M_{\pi_i},\psi}(\psi(k) = \pi_j)$ with $1 \leqslant i, j \leqslant (d-1)(H-1)$ to denote the probability that ψ takes policy π_j at deployment k when running ψ on M_{π_i}, and use $P_{M_{\pi_i},\psi}(\psi(K+1) = \pi_i)$ to denote the probability that the algorithm ψ returns policy π_i as optimal arm after running with K deployments under MDP M_{π_i}. We are interested in providing an upper bound for the expected success rate:

$$
\begin{aligned}
P_{\psi, M\sim\mathcal{M}}(\text{success}) :=& \frac{1}{|\mathcal{M}|} P(\text{success in } M_0) + \frac{1}{|\mathcal{M}|} \sum_{i=1}^{|\bar{\mathcal{M}}|} P_{M_{\pi_i},\psi}(\psi(K+1) = \pi_i) \\
=& \frac{1}{|\mathcal{M}|} + \frac{1}{|\mathcal{M}|} \sum_{i=1}^{|\bar{\mathcal{M}}|} P_{M_{\pi_i},\psi}(\psi(K+1) = \pi_i),
\end{aligned}
$$

where we assume that all the policies in M_0 are optimal policies.

In the following, we use E_{k,π_i} to denote the event that the policy π_i has been deployed at least once in the first k deployments and $P_{M_{\pi_j},\psi}(\cdot)$ to denote the probability of an event when running algorithm ψ under MDP M_{π_j}.

Next, we prove that, for arbitrary M_{π_i},

$$
P_{M_{\pi_i},\psi}(E_{k,\pi_i}^{\complement}) = P_{M_0,\psi}(E_{k,\pi_i}^{\complement}) \quad \forall k \in [K+1]. \tag{4}
$$

First of all, it holds for $k = 1$, because at the beginning ψ has't observe any data, and all its possible behavior should be the same in both M_0 and M_i, and therefore $P_{M_{\pi_i},\psi}(E_{1,\pi_i}^{\complement}) = P_{M_0,\psi}(E_{1,\pi_i}^{\complement})$. Next, we do induction. Suppose we already know it holds for $1, 2, ..., k$, then consider the case for $k+1$. Because ψ behave the same if the pre-collected episodes are the same, which is the only information it will use for decision, we should have:

$$
\begin{aligned}
P_{M_{\pi_i},\psi}(\psi(k+1) = \pi_i \cap E_{k,\pi_i}^{\complement}) =& \sum_{\tau \in \psi(k+1) = \pi_i \cap E_{k,\pi_i}^{\complement}} P_{M_{\pi_i},\psi}(\tau) \\
=& \sum_{\tau \in \psi(k+1) = \pi_i \cap E_{k,\pi_i}^{\complement}} P_{M_0,\psi}(\tau)
\end{aligned}
$$

$$=P_{M_0,\psi}(\psi(k+1)=\pi_i \cap E^{\complement}_{k,\pi_i}). \tag{5}$$

The second equality is due to each trajectory $\tau \in \psi(k+1)=\pi_i \cap E^{\complement}_{k,\pi_i}$ has the same probability under M_0 and M_i by the construction. Notice that in this induction step, we only consider the trajectory with first $(k+1)N$ episodes because define the whole sample space and event only based on the first $(k+1)N$ episodes.

This implies that,

$$\begin{aligned}
P_{M_{\pi_i},\psi}(E^{\complement}_{k+1,\pi_i}) =& P_{M_{\pi_i},\psi}(E^{\complement}_{k,\pi_i}) - P_{M_{\pi_i},\psi}(\psi(k+1)=\pi_i \cap E^{\complement}_{k,\pi_i}) \\
=& P_{M_0,\psi}(E^{\complement}_{k,\pi_i}) - P_{M_0,\psi}(\psi(k+1)=\pi_i \cap E^{\complement}_{k,\pi_i}) \\
=& P_{M_0,\psi}(E^{\complement}_{k+1,\pi_i}).
\end{aligned}$$

Now we are ready to bound the failure rate. Suppose $K < (d-1)(H-1)/2 < |\mathcal{M}|/2$, we have:

$$\begin{aligned}
&\frac{1}{|\mathcal{M}|}\sum_{i=1}^{|\bar{\mathcal{M}}|}\left(P_{M_{\pi_i},\psi}(\psi(K+1)=\pi_i) - P_{M_0,\psi}(\psi(K+1)=\pi_i)\right) \\
=&\frac{1}{|\mathcal{M}|}\sum_{i=1}^{|\bar{\mathcal{M}}|}\Big(P_{M_{\pi_i},\psi}(\psi(K+1)=\pi_i \cap E_{K+1,\pi_i}) - P_{M_0,\psi}(\psi(K+1)=\pi_i \cap E_{K+1,\pi_i})\Big) \\
&+\frac{1}{|\mathcal{M}|}\sum_{i=1}^{|\bar{\mathcal{M}}|}\Big(P_{M_{\pi_i},\psi}(\psi(K+1)=\pi_i \cap E^{\complement}_{K+1,\pi_i}) - P_{M_0,\psi}(\psi(K+1)=\pi_i \cap E^{\complement}_{K+1,\pi_i})\Big) \\
=&\frac{1}{|\mathcal{M}|}\sum_{i=1}^{|\bar{\mathcal{M}}|}P_{M_{\pi_i},\psi}(E_{K+1,\pi_i})\Big(P_{M_{\pi_i},\psi}(\psi(K+1)=\pi_i|E_{K+1,\pi_i}) - P_{M_0,\psi}(\psi(K+1)=\pi_i|E_{K+1,\pi_i})\Big) \\
&\qquad\qquad \text{(Eq.(5) and } P(A\cap B)=P(B)P(A|B)) \\
\leqslant&\frac{1}{|\mathcal{M}|}\sum_{i=1}^{|\bar{\mathcal{M}}|}P_{M_{\pi_i},\psi}(E_{K+1,\pi_i}) \\
&\qquad\qquad (P_{M_{\pi_i},\psi}(\psi(K+1)=\pi_i|E_{K+1,\pi_i}) - P_{M_0,\psi}(\psi(K+1)=\pi_i|E_{K+1,\pi_i}) \leqslant 1) \\
=&\frac{1}{|\mathcal{M}|}\sum_{i=1}^{|\bar{\mathcal{M}}|}P_{M_0,\psi}(E_{K+1,\pi_i}) \qquad\qquad \text{(Eq. (4))} \\
\leqslant&\frac{K}{|\mathcal{M}|}
\end{aligned}$$

where the last step is because:

$$\begin{aligned}
&\frac{1}{|\mathcal{M}|}\sum_{i=1}^{|\bar{\mathcal{M}}|}P_{M_0,\psi}(E_{K+1,\pi_i}) \\
=&\frac{1}{|\mathcal{M}|}\sum_{i=1}^{|\bar{\mathcal{M}}|}\mathbb{E}_{M_0,\psi}[\mathbf{1}\{\pi_i \text{ is selected}\}] \\
\leqslant&\frac{1}{|\mathcal{M}|}\sum_{i=1}^{|\bar{\mathcal{M}}|}\mathbb{E}_{M_0,\psi}[\sum_{k=1}^{K+1}\mathbf{1}\{\pi_i \text{ is selected at deployment } k\}] \\
=&\frac{1}{|\mathcal{M}|}\mathbb{E}_{M_0,\psi}[\sum_{k=1}^{K+1}\sum_{i=1}^{|\bar{\mathcal{M}}|}\mathbf{1}\{\pi_i \text{ is selected at deployment } k\}] \\
=&\frac{K+1}{|\mathcal{M}|} \qquad\qquad (\psi \text{ deploy deterministic policy each time}) \\
\leqslant&\frac{1}{2} \qquad\qquad (\text{Deployment time } K < |\mathcal{M}|/2)
\end{aligned}$$

As a result,

$$\frac{1}{|\mathcal{M}|}+\frac{1}{|\mathcal{M}|}\sum_{i=1}^{|\bar{\mathcal{M}}|}P_{M_{\pi_i},\psi}(\psi(K+1)=\pi_i)\leqslant\frac{1}{|\mathcal{M}|}+\frac{1}{|\mathcal{M}|}\sum_{i=1}^{|\bar{\mathcal{M}}|}P_{M_0,\psi}(\psi(K+1)=\pi_i)+\frac{1}{2}\leqslant\frac{2}{|\mathcal{M}|}+\frac{1}{2}.$$

As long as $d, H \geqslant 3$, we have $|\mathcal{M}| = (d-1)(H-1)+1 \geqslant 5$ the failure rate will be higher than $1/10$, which finishes the proof. □

B.3 Proof for Lower Bound in Arbitrary Setting

In the following, we provide a formal statement of the lower bound theorem for the arbitrary policy setting and its proof.

Theorem B.3 (Lower bound for number of deployments in arbitrary setting)**.** *For the linear MDP problem with given dimension $d \geqslant 2$ and horizon $H \geqslant 3$, $N = \mathrm{poly}(d, H, \frac{1}{\varepsilon}, \log\frac{1}{\delta})$, and arbitrary given algorithm ψ. Unless the number of deployments $K > \frac{H-2}{2\lceil\log_d NH\rceil} = \Omega(H/\log_d(NH)) = \widetilde{\Omega}(H)$, for any ε, there exists an MDP such that the output policy is not ε-optimal with probability at least $\frac{1}{2e}$. Here ψ can be deterministic or stochastic. The algorithm can deploy arbitrary policy but can only collect $N = \mathrm{poly}(d, H, \frac{1}{\varepsilon}, \log\frac{1}{\delta})$ samples in each deployment.*

Proof. Since any randomized algorithm is just a distribution over deterministic ones, it suffices to only consider deterministic algorithms ψ in the following proof (Krishnamurthy et al., 2016). The crucial part here is notice that a deployment means we have a fixed distribution (occupancy) over the state space and such distribution only depends on the prior information.

Construction of Hard Instances We have $d^{H-2} \times d$ instances by enumerating the location of core state from level 1 to $H-2$ and the optimal normal state at level $H-1$. We assign $s_{H-1}^{(i+1)\%d}$ as the core state at level $H-1$ if s_{H-1}^{i} is the optimal state. Notice that for this hard instance class, we only consider the case that the optimal state is in level $H-1$. We use $\mathcal{M}$ to denote this hard instance class.

We make a few claims and later prove these claims and the theorem. We will use the notation $E_i(j)$ to denote the event that at least one state at level j is reached by the i-th deployment. Also we notice that in all the discussion an event is just a set of trajectories. For all related discussion, the state at level L does not include the state in the absorbing chain. In addition, we will use $P_{\mathcal{M},\psi}$ to denote the distribution of trajectories when executing algorithm ψ and uniformly taking an instance from the hard instance class.

Claim 1. *Assume $L \leqslant H-2$. Then for any deterministic algorithm ψ, we have*

$$P_{\mathcal{M},\psi}(E_1(L)) \leqslant \frac{N}{d^{L-1}}.$$

Claim 2. *Assume $L + L' \leqslant H-2$. We have that for any deterministic algorithm ψ,*

$$P_{\mathcal{M},\psi}(E_k^{\complement}(L'+L)) \geqslant (1-\frac{N}{d^{L-1}})P(E_{k-1}^{\complement}(L')).$$

Proof of Claim 1 By the nature of the deterministic algorithm, we know that for any deterministic algorithm ψ, the deployment is the same at the first time for all instances. The reason is that the agent hasn't observed anything, so the deployed policy has to be the same.

Let $p(i, j, h)$ denote the probability of the first deployment policy to choose action j at node i at level h under the first deployment policy. We know that $p(i, j, h)$ for ψ is the same under all instances.

Note that there is a one to one correspondence between an MDP in the hard instance class and the specified locations of core states in the layer $1, \ldots, H-2$ and the optimal state at level $H-1$. Therefore, we can use $(i_1, \ldots, i_{H-2}, s_{H-1})$ to denote any instance in the hard instance class, where $i_1, \ldots, i_{H-2}$ refers to the location of the core states and s_{H-1} refers to the location of the optimal state. From the construction, we know that for instance $(i_1, \ldots, i_{H-2}, s_{H-1})$, to arrive at level s_L

at level L, the path as to be $s_0, i_1, \ldots, i_{L-1}, s_L$. Therefore the probability of a trajectory sampled from ψ to reach state s_L is

$$p(s_0, i_1, 0)p(i_1, i_2, 1)\ldots p(i_{L-2}, i_{L-1}, L-2)p(i_{L-1}, s_L, L-1).$$

Here we use $p(i_{L-1}, s_L, L-1)$ denotes the probability of taking action at i_{L-1} to transit to s_L and similarly for others. In the deployment, ψ draws N episodes, so the probability of executing ψ to reach any state s_L at level L during the first deployment is no more than $Np(s_0, i_1, 0)p(i_1, i_2, 1)\ldots p(i_{L-2}, i_{L-1}, L-2)p(i_{L-1}, s_L, L-1)$.

Calculating the sum over $i_1, \ldots, i_{L-1}$ and s_L gives us

$$\begin{aligned}
&\sum_{i_1, i_2, \ldots, i_{L-1}, s_L} Np(s_0, i_1, 0)p(i_1, i_2, 1)\ldots p(i_{L-2}, i_{L-1}, L-2)p(i_{L-1}, s_L, L-1)\\
=&N\sum_{i_1} p(s_0, i_1, 0)\sum_{i_2} p(i_1, i_2, 1)\ldots \sum_{i_{L-2}} p(i_{L-3}, i_{L-2}, L-3)\sum_{i_{L-1}} p(i_{L-2}, i_{L-1}, L-2)\sum_{s_L} p(i_{L-1}, s_L, L-1)\\
=&N\sum_{i_1} p(s_0, i_1, 0)\sum_{i_2} p(i_1, i_2, 1)\ldots \sum_{i_{L-2}} p(i_{L-3}, i_{L-2}, L-3)\sum_{i_{L-1}} p(i_{L-2}, i_{L-1}, L-2)\\
=&N\sum_{i_1} p(s_0, i_1, 0)\sum_{i_2} p(i_1, i_2, 1)\ldots \sum_{i_{L-2}} p(i_{L-3}, i_{L-2}, L-3)\\
=&\ldots\\
=&N. \qquad (6)
\end{aligned}$$

Therefore we have the following equation about $P_{\mathcal{M},\psi}(E_1(L))$

$$\begin{aligned}
&P_{\mathcal{M},\psi}(E_1(L))\\
=&\frac{1}{|\mathcal{M}|}\sum_{i_L, \ldots, i_{H-2}, s_{H-1}}\sum_{i_1, \ldots, i_{L-1}} P_{I=(i_1, \ldots, i_{H-1}, s_{H-1}), \psi}(E_1(L))\\
\leqslant&\frac{1}{d^{H-1}}\sum_{i_L, \ldots, i_{H-2}, s_{H-1}}\sum_{i_1, \ldots, i_{L-1}, s_L} Np(s_0, i_1, 0)p(i_1, i_2, 1)\ldots p(i_{L-2}, i_{L-1}, L-2)p(i_{L-1}, s_L, L-1)\\
=&\frac{1}{d^{H-1}}\sum_{i_L, \ldots, i_{H-2}, s_{H-1}} N\\
=&\frac{N}{d^{L-1}}.
\end{aligned}$$

Proof of Claim 2 Let τ denote any possible concatenation of the first kN episodes we get in the first k deployments. In this claim, it suffices to consider the kN episodes because the event $E_k(L'+L) \cap E_{k-1}^{\complement}(L')$ only depends on the first kN episodes. Therefore the sample space and the event will be defined on any trajectory with kN episodes. For any τ, we know that ψ will output the k-th deployment policy solely based on the $\tau[0, k-1]$ and this map is deterministic (we use $\tau[i, j]$ to denote the $iN+1$ to jN episodes in τ). In other words, ψ will map $\tau[0, k-1]$ to a fixed policy $\psi(\tau[0, k-1])$ to deploy at the k-th time.

We have the following equation for any $I \in \mathcal{M}$

$$\begin{aligned}
&P_{I,\psi}(E_k(L'+L) \cap E_{k-1}^{\complement}(L'))\\
=&\sum_{\tau \in E_k(L'+L) \cap E_{k-1}^{\complement}(L')} P_{I,\psi}(\tau)\\
=&\sum_{\tau \in E_k(L'+L) \cap E_{k-1}^{\complement}(L')} P_{I,\psi}(\tau[0, k-1])P_{I,\psi(\tau[0,k-1])}(\tau[k-1, k])\\
=&\sum_{\tau:\tau[0,k-1] \in E_{k-1}^{\complement}(L'), \tau':\tau'[0,k-1]=\tau[0,k-1] \text{ and } \tau'[k-1,k] \text{ hit level } L'+L} P_{I,\psi}(\tau[0, k-1])P_{I,\psi(\tau[0,k-1])}(\tau'[k-1, k])\\
=&\sum_{\tau:\tau[0,k-1] \in E_{k-1}^{\complement}(L')} P_{I,\psi}(\tau[0, k-1])\sum_{\tau':\tau'[0,k-1]=\tau[0,k-1] \text{ and } \tau'[k-1,k] \text{ hit level } L'+L} P_{I,\psi(\tau[0,k-1])}(\tau'[k-1, k])
\end{aligned}$$

Notice that this equality does not generally hold for probability distribution $P_{\mathcal{M},\psi}$.

Then we fix $\tau[0,k-1]$, such that $\tau[0,k-1] \in E^{\complement}_{k-1}(L')$. We also fix $(i_1,\dots,i_{L'-1})$, $(i_{L'+L},\dots,i_{H-2},s_{H-1})$ and consider two instances $I_1 = (i_1,\dots,i_{L'-1},i^1_{L'},\dots,i^1_{L'+L-1},i_{L'+L},\dots,i_{H-2},s_{H-1})$ and $I_2 = (i_1,\dots,i_{L'-1},i^2_{L'},\dots,i^2_{L'+L-1},i_{L'+L},\dots,i_{H-2},s_{H-1})$. Therefore, we have that $P_{I_1,\psi}(\tau[0,k-1]) = P_{I_2,\psi}(\tau[0,k-1])$ (from the construction of I_1, I_2 and the property of deterministic algorithm ψ). We use $I(i_1,\dots,i_{L'-1},i_{L'+L},\dots,i_{H-2},s_{H-1})$ to denote the instance class that has fixed $(i_1,\dots,i_{L'-1})$, $(i_{L'+L},\dots,i_{H-2},s_{H-1})$, but different $(i_{L'},\dots i_{L'+L-1})$. In addition, we use $I(i_1,\dots,i_{L'-1})$ to denote the instance class that has fixed $(i_1,\dots,i_{L'-1})$, but different $(i_{L'},\dots i_{L'+L-1})$ and $(i_{L'+L},\dots,i_{H-2},s_{H-1})$.

Since we have already fixed $\tau[0,k-1] \in E^{\complement}_{k-1}(L')$ here, $\psi(\tau[0,k-1])$ is also fixed (for all $I \in I(i_1,\dots,i_{L'-1},i_{L'+L},\dots,i_{H-2},s_{H-1})$). Also notice that we are considering the probability of N episodes $\tau'[k-1:k]$. Therefore, we can follow Claim 1 and define $p(i,j,h)$ for $0 \leqslant h \leqslant L'+L-1$, which represents the probability of choosing action j at node i at level h under the k-th deployment policy. In the k-th deployment, ψ draws N episodes, so the probability of executing ψ to reach any state $s_{L'+L}$ at level $L'+L$ under instance $I = (i_1,\dots,i_{H-2},s_{H-1})$ is

$$
\begin{aligned}
&Np(s_0,i_1,0)p(i_1,i_2,1)\dots p(i_{L'+L-2},i_{L'+L-1},L'+L-2)p(i_{L'+L-1},s_{L'+L},L'+L-1)\\
\leqslant &Np(i_{L'-1},i_{L'},L'-1)\dots p(i_{L'+L-2},i_{L'+L-1},L'+L-2)p(i_{L'+L-1},s_{L'+L},L'+L-1).
\end{aligned}
$$

Following the same step in Eq (6) by summing over $i_{L'},\dots,i_{L'+L-1}$ and $s_{L'+L}$ gives us

$$
\begin{aligned}
&\sum_{I\in I(i_1,\dots,i_{L'-1},i_{L'+L},\dots,i_{H-2},s_{H-1})} \sum_{\tau':\tau'[0,k-1]=\tau[0,k-1] \text{ and } \tau'[k-1,k] \text{ hit level } L'+L} P_{I,\psi(\tau[0,k-1])}(\tau'[k-1,k])\\
\leqslant &N.
\end{aligned}
$$

Now, we sum over all possible $(i_{L'+L},\dots,i_{H-2},s_{H-1})$ and take the average. For any fixed $\tau[0,k-1] \in E^{\complement}_{k-1}(L')$ we have

$$
\begin{aligned}
&\frac{1}{|I(i_1,\dots,i_{L'-1})|}\sum_{I\in I(i_1,\dots,i_{L'-1})} \sum_{\tau':\tau'[0,k-1]=\tau[0,k-1] \text{ and } \tau'[k-1,k] \text{ hit level } L'+L} P_{I,\psi(\tau[0,k-1])}(\tau[k-1,k])\\
=&\frac{1}{|I(i_1,\dots,i_{L'-1})|}\sum_{i_{L'+L},\dots,i_{H-2},s_{H-1}} \sum_{I\in I(i_1,\dots,i_{L'-1},i_{L'+L},\dots,i_{H-2},s_{H-1})}\\
&\sum_{\tau':\tau'[0,k-1]=\tau[0,k-1] \text{ and } \tau'[k-1,k] \text{ hit level } L'+L} P_{I,\psi(\tau[0,k-1])}(\tau[k-1,k])\\
\leqslant&\frac{1}{|I(i_1,\dots,i_{L'-1})|}\sum_{i_{L'+L},\dots,i_{H-2},s_{H-1}} N\\
=&\frac{d^{H-L'-L}}{d^{H-L'}}N\\
=&\frac{1}{d^L}N.
\end{aligned}
$$

Moreover, summing over all $\tau[0,k-1] \in E^{\complement}_{k-1}(L')$, gives us $i_1,\dots,i_{L'-1}$

$$
\begin{aligned}
&P_{\mathcal{M},\psi}(E_k(L'+L)\cap E^{\complement}_{k-1}(L'))\\
=&\frac{1}{|\mathcal{M}|}\sum_{i_1,\dots,i_{L'-1}} \sum_{I\in I(i_1,\dots,i_{L'-1})} \sum_{\tau[0,k-1]\in E^{\complement}_{k-1}(L')} P_{I,\psi}(\tau[0,k-1])\\
&\sum_{\tau':\tau'[0,k-1]=\tau[0,k-1] \text{ and } \tau'[k-1,k] \text{ hit level } L'+L} P_{I,\psi(\tau[0,k-1])}(\tau'[k-1,k])
\end{aligned}
$$

$$
\begin{aligned}
&= \frac{1}{|\mathcal{M}|} \sum_{\tau[0,k-1] \in E^{\complement}_{k-1}(L')} \sum_{i_1,\ldots,i_{L'-1}} P_{i_1,\ldots,i_{L'-1},\psi}(\tau[0,k-1]) \\
&\qquad \sum_{I \in I(i_1,\ldots,i_{L'-1})} \sum_{\tau' : \tau'[0,k-1] = \tau[0,k-1] \text{ and } \tau'[k-1,k] \text{ hit level } L'+L} P_{I,\psi(\tau[0,k-1])}(\tau'[k-1,k]) \\
&\leqslant \frac{1}{|\mathcal{M}|} \sum_{\tau[0,k-1] \in E^{\complement}_{k-1}(L')} \sum_{i_1,\ldots,i_{L'-1}} P_{i_1,\ldots,i_{L'-1},\psi}(\tau[0,k-1]) |I(i_1,\ldots,i_{L'-1})| \frac{N}{d^L} \\
&= \frac{N}{d^L} \frac{|I(i_1,\ldots,i_{L'-1})|}{|\mathcal{M}|} \sum_{\tau[0,k-1] \in E^{\complement}_{k-1}(L')} \sum_{i_1,\ldots,i_{L'-1}} P_{i_1,\ldots,i_{L'-1},\psi}(\tau[0,k-1]) \\
&= \frac{N}{d^L} \sum_{\tau[0,k-1] \in E^{\complement}_{k-1}(L')} P_{\mathcal{M},\psi}(\tau[0,k-1]) \\
&= \frac{N}{d^L} P_{\mathcal{M},\psi}(E^{\complement}_{k-1}(L')).
\end{aligned}
$$

In the second equality, we use $P_{i_1,\ldots,i_{L'-1},\psi}$ because for any fixed $\tau[0,k-1] \in E^{\complement}_{k-1}(L')$ and all $I \in I(i_1,\ldots,i_{L'-1})$, $P_{i_1,\ldots,i_{L'-1},\psi}(\tau[0,k-1])$ are the same.

Finally, we have

$$
\begin{aligned}
P_{\mathcal{M},\psi}(E^{\complement}_k(L'+L)) &\geqslant P_{\mathcal{M},\psi}(E^{\complement}_k(L'+L) \cap E^{\complement}_{k-1}(L')) \\
&= P_{\mathcal{M},\psi}(E^{\complement}_{k-1}(L')) - P_{\mathcal{M},\psi}(E_k(L'+L) \cap E^{\complement}_{k-1}(L')) \\
&\geqslant (1 - \frac{N}{d^L}) P(E^{\complement}_{k-1}(L')) \\
&\geqslant (1 - \frac{N}{d^{L-1}}) P(E^{\complement}_{k-1}(L')).
\end{aligned}
$$

Proof of the Theorem If $KL \leqslant H-2$, then applying Claim 2 for $K-1$ times and applying Claim 1 tells us

$$
\begin{aligned}
P_{\mathcal{M},\psi}(E^{\complement}_K(KL)) =& P_{\mathcal{M},\psi}(E^{\complement}_K((K-1)L+L)) \geqslant (1 - \frac{N}{d^{L-1}}) P(E^{\complement}_{K-1}((K-1)L)) \\
&\geqslant \ldots \geqslant (1 - \frac{N}{d^{L-1}})^{K-1} P(E^{\complement}_1(L)) \geqslant (1 - \frac{N}{d^{L-1}})^K.
\end{aligned}
$$

We can set $L = \lceil \log_d NH \rceil + 1$ and $K \leqslant \frac{H-2}{2\lceil \log_d NH \rceil}$. Then for $H \geqslant 3$, we get $KL \leqslant H-2$ and

$$
\begin{aligned}
P_{\mathcal{M},\psi}(\text{does not hit any state at level } H-2) &\geqslant (1 - \frac{N}{d^{\lceil \log_d NH \rceil}})^{\frac{H-2}{2\lceil \log_d NH \rceil}} \\
&\geqslant (1 - \frac{N}{NH})^{\frac{H-2}{2\lceil \log_d NH \rceil}} \\
&\geqslant (1 - \frac{1}{H})^H \\
&\geqslant \frac{1}{e}.
\end{aligned}
$$

Let event F denote the event (a set of length KN episodes trajectories) that any state at level $H-2$ is not hit. Then we have $P_{\mathcal{M},\psi}(F) \geqslant 1 - \frac{1}{e}$. We use $I(i_1,\ldots,i_{H-2})$ to denote the instance class that has fixed core states $(i_1,\ldots,i_{H-2})$ but different optimal states s_{H-1}.

Consider any fixed $\tau \in F$. Similar as the proof in the prior claims, by the property of deterministic algorithm, we can define $p(i,j,h)$ for $h = H-2$, which represents the probability of the output policy $\psi_\tau(K+1)$ under trajectory τ to choose action j at node i at level $H-2$. Then we have

$$
\sum_{I=(i_1,\ldots,i_{H-2},s_{H-1}) \in I(i_1,\ldots,i_{H-2})} P_{I,\psi}(\psi_\tau(K+1) \text{ chooses optimal state})
$$

$$
\begin{aligned}
&= \sum_{I=(i_1,\ldots,i_{H-2},s_{H-1})\in I(i_1,\ldots,i_{H-2})} P_{I,\psi}((\psi_\tau(K+1))(i_{H-2}) = s_{H-1}) \\
&= \sum_{I=(i_1,\ldots,i_{H-2},s_{H-1})\in I(i_1,\ldots,i_{H-2})} p(i_{H-2}, s_{H-1}, H-2) \\
&= \sum_{s_{H-1}} p(i_{H-2}, s_{H-1}, H-2) \\
&=1.
\end{aligned}
$$

Summing over $\tau \in F$ gives us

$$
\begin{aligned}
&\sum_{I=(i_1,\ldots,i_{H-2},s_{H-1})\in I(i_1,\ldots,i_{H-2})} P_{I,\psi}(F \cap \text{ the output policy chooses optimal state}) \\
=&\sum_{I=(i_1,\ldots,i_{H-2},s_{H-1})\in I(i_1,\ldots,i_{H-2})} \sum_{\tau\in F} P_{I,\psi}(\tau) P_{I,\psi}(\psi_\tau(K+1) \text{ chooses optimal state}) \\
=&\sum_{I=(i_1,\ldots,i_{H-2},s_{H-1})\in I(i_1,\ldots,i_{H-2})} \sum_{\tau\in F} P_{i_1,\ldots,i_{H-2},\psi}(\tau) P_{I,\psi}(\psi_\tau(K+1) \text{ chooses optimal state}) \\
=&\sum_{\tau\in F} P_{i_1,\ldots,i_{H-2},\psi}(\tau) \sum_{I=(i_1,\ldots,i_{H-2},s_{H-1})\in I(i_1,\ldots,i_{H-2})} P_{I,\psi}(\psi_\tau(K+1) \text{ chooses optimal state}) \\
=&\sum_{\tau\in F} P_{i_1,\ldots,i_{H-2},\psi}(\tau) \\
=&P_{i_1,\ldots,i_{H-2},\psi}(F).
\end{aligned}
$$

In the second equality, we notice that for all instance $I \in I = (i_1, \ldots, i_{H-2}, s_{H-1})$, $P_{I,\psi}(\tau)$ are the same, so this probability distribution essentially depends on $i_1, \ldots, i_{H-2}$. In the third inequality, we change the order of the summation.

Finally, summing over $i_1, \ldots, i_{H-2}$ and taking average yields that

$$
\begin{aligned}
&P_{\mathcal{M},\psi}(F \cap \text{ the output policy chooses optimal state}) \\
=&\frac{1}{d^{H-1}} \sum_{i_1,\ldots,i_{H-2}} \sum_{I=(i_1,\ldots,i_{H-2},s_{H-1})\in I(i_1,\ldots,i_{H-2})} P_{I,\psi}(F \cap \text{ the output policy chooses optimal state}) \\
=&\frac{1}{d^{H-1}} \sum_{i_1,\ldots,i_{H-2}} P_{i_1,\ldots,i_{H-2},\psi}(F) \\
=&\frac{1}{d^{H-1}} \sum_{i_1,\ldots,i_{H-2}} \frac{1}{d} \sum_{s_{H-1}} P_{I=(i_1,\ldots,i_{H-2},s_{H-1}),\psi}(F) \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (P_{I,\psi}(F) \text{ does not depend on the optimal state}) \\
=&\frac{1}{d} P_{\mathcal{M},\psi}(F)
\end{aligned}
$$

Therefore, we get the probability of not choosing the optimal state is

$$
\begin{aligned}
&P_{\mathcal{M},\psi}(\text{ the output policy does not choose the optimal state}) \\
\geqslant&P_{\mathcal{M},\psi}(F \cap \text{ the output policy does not choose the optimal state}) \\
=&P_{\mathcal{M},\psi}(F) - P_{\mathcal{M},\psi}(F \cap \text{ the output policy chooses the optimal state}) \\
=&\frac{d-1}{d} P_{\mathcal{M},\psi}(F) \\
\geqslant&\frac{1}{2} \cdot \frac{1}{e}.
\end{aligned}
$$

From the construction, we know that any policy that does not choose optimal state (thus also does not choose the optimal action associated with the optimal state) is ε sub-optimal. This implies that with probability at least $\frac{1}{2e}$, the output policy is at least ε sub-optimal. □

C DEPLOYMENT-EFFICIENT RL WITH DETERMINISTIC POLICIES AND GIVEN REWARD FUNCTION

C.1 ADDITIONAL NOTATIONS

In the appendix, we will frequently consider the MDP truncated at $\widetilde{h} \leqslant H$, and we will use:

$$V_h^\pi(s|\widetilde{h}) = \mathbb{E}[\sum_{h'=h}^{\widetilde{h}} r_{h'}(s_{h'}, a_{h'})|s_h = s, \pi], \quad Q_h^\pi(s,a|\widetilde{h}) = \mathbb{E}[\sum_{h'=h}^{\widetilde{h}} r_{h'}(s_{h'}, a_{h'})|s_h = s, a_h = a, \pi]$$

to denote the value function in truncated MDP for arbitrary $h \leqslant \widetilde{h}$, and also extend the definition in Section 2 to $V_h^*(\cdot|\widetilde{h}), Q_h^*(\cdot,\cdot|\widetilde{h}), \pi^*_{|\widetilde{h}}$ for optimal policy setting and $V_h^*(\cdot, r|\widetilde{h}), Q_h^*(\cdot,\cdot,r|\widetilde{h}), \pi^*_{r|h}$ for reward-free setting.

C.2 AUXILIARY LEMMA

Lemma C.1 (Elliptical Potential Lemma; Lemma 26 of Agarwal et al. (2020b))**.** *Consider a sequence of $d \times d$ positive semi-definite matrices $X_1, ..., X_T$ with $\max_t Tr(X_t) \leqslant 1$ and define $M_0 = \lambda I, ..., M_t = M_{t-1} + X_t$. Then*

$$\sum_{t=1}^{T} Tr(X_t M_{t-1}^{-1}) \leqslant (1 + 1/\lambda) d \log(1 + T/d).$$

Lemma C.2 (Abbasi-yadkori et al. (2011))**.** *Suppose $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{d\times d}$ are two positive definite matrices satisfying $\mathbf{A} \geq \mathbf{B}$, then for any $x \in \mathbb{R}^d$, we have:*

$$\|x\|_{\mathbf{A}}^2 \leqslant \|x\|_{\mathbf{B}}^2 \frac{\det(\mathbf{A})}{\det(\mathbf{B})}.$$

Next, we prove a lemma to bridge between trace and determinant, which is crucial to prove our key technique in Lemma 4.2.

Lemma C.3. *[Bridge between Trace and Determinant] Consider a sequence of matrices $\mathbf{A}_0, \mathbf{A}_N, ..., \mathbf{A}_{(K-1)N}$ with $\mathbf{A}_0 = I$ and $\mathbf{A}_{kN} = \mathbf{A}_{(k-1)N} + \Phi_{k-1}$, where $\Phi_{k-1} = \sum_{t=(k-1)N+1}^{kN} \phi_t \phi_t^\top$. We have*

$$Tr(\mathbf{A}_{(k-1)N}^{-1} \Phi_{k-1}) \leqslant \frac{\det(\mathbf{A}_{kN})}{\det(\mathbf{A}_{(k-1)N})} \log \frac{\det(\mathbf{A}_{kN})}{\det(\mathbf{A}_{(k-1)N})}.$$

Proof. Consider a more general case, given matrix $\mathbf{Y} \geq I$, we have the following inequality

$$Tr(\mathbf{I} - \mathbf{Y}^{-1}) \leqslant \log\det(\mathbf{Y}) \leqslant Tr(\mathbf{Y} - \mathbf{I}).$$

By replacing $\mathbf{Y}$ with $\mathbf{I} + \mathbf{A}^{-1}\mathbf{X}$ in the above inequality, we have:

$$\begin{aligned} Tr((\mathbf{A}+\mathbf{X})^{-1}\mathbf{X}) =& Tr((\mathbf{I}+\mathbf{A}^{-1}\mathbf{X})^{-1}(\mathbf{A}^{-1}\mathbf{X})) = Tr((\mathbf{I}+\mathbf{A}^{-1}\mathbf{X})^{-1}(\mathbf{I}+\mathbf{A}^{-1}\mathbf{X}-\mathbf{I}) \\ =& Tr(\mathbf{I} - (\mathbf{I}+\mathbf{A}^{-1}\mathbf{X})^{-1}) \\ \leqslant& \log\det(\mathbf{I}+\mathbf{A}^{-1}\mathbf{X}) = \log\frac{\det(\mathbf{A}+\mathbf{X})}{\det(\mathbf{A})}. \end{aligned}$$

By assigning $\mathbf{A} = \mathbf{A}_{(k-1)N}$ and $\mathbf{X} = \Phi_{k-1}$, and applying Lemma C.2, we have:

$$\begin{aligned} Tr(\mathbf{A}_{(k-1)N}^{-1}\Phi_{k-1}) =& \sum_{t=(k-1)N+1}^{kN} \|\phi_t\|^2_{\mathbf{A}_{(k-1)N}^{-1}} \\ \leqslant& \sum_{t=(k-1)N+1}^{kN} \|\phi_t\|^2_{\mathbf{A}_{kN}^{-1}} \frac{\det \mathbf{A}_{kN}}{\det(\mathbf{A}_{(k-1)N})} \end{aligned}$$

$$
\begin{aligned}
&= Tr(\mathbf{A}_{kN}^{-1}\Phi_{k-1})\frac{\det \mathbf{A}_{kN}}{\det(\mathbf{A}_{(k-1)N})} \\
&\leqslant \frac{\det \mathbf{A}_{kN}}{\det(\mathbf{A}_{(k-1)N})}\log\frac{\det \mathbf{A}_{kN}}{\det(\mathbf{A}_{(k-1)N})}
\end{aligned}
$$

which finished the proof. □

Lemma 4.2. *[Finite Sample Elliptical Potential Lemma] Consider a sequence of matrices* $\mathbf{A}_0, \mathbf{A}_N, ..., \mathbf{A}_{(K-1)N} \in \mathbb{R}^{d\times d}$ *with* $\mathbf{A}_0 = I_{d\times d}$ *and* $\mathbf{A}_{kN} = \mathbf{A}_{(k-1)N} + \Phi_{k-1}$*, where* $\Phi_{k-1} = \sum_{t=(k-1)N+1}^{kN}\phi_t\phi_t^\top$ *and* $\max_{t\leqslant KN}\|\phi_t\| \leqslant 1$*. We define:* $\mathcal{K}^+ := \Big\{k \in [K]\Big| Tr(\mathbf{A}_{(k-1)N}^{-1}\Phi_{k-1}) \geqslant N\varepsilon\Big\}$*. For arbitrary* $\varepsilon < 1$*, and arbitrary* $c_K \geqslant 2$*, if* $K = c_K dH + 1$*, by choosing* $N \geqslant c\Big(c_K\frac{Hd^{c_K}}{\varepsilon^{c_K}}\log^{c_K}(\frac{Hd}{\varepsilon})\Big)^{\frac{1}{c_K-1}}$*, where* c *is an absolute constant independent with* c_K, d, H, ε*, we have* $|\mathcal{K}^+| \leqslant c_K d < K/H$*.*

Proof. Suppose we have $Tr(\mathbf{A}_{(k-1)N}^{-1}\Phi_{k-1}) \geqslant N\varepsilon$, by applying Lemma C.3 we must have:

$$
\begin{aligned}
N\varepsilon &\leqslant \frac{\det(\mathbf{A}_{kN})}{\det(\mathbf{A}_{(k-1)N})}\log\frac{\det(\mathbf{A}_{kN})}{\det(\mathbf{A}_{(k-1)N})} \leqslant \frac{\det(\mathbf{A}_{kN})}{\det(\mathbf{A}_{(k-1)N})}\log(\det(\mathbf{A}_{kN})) \\
&\leqslant d\frac{\det(\mathbf{A}_{kN})}{\det(\mathbf{A}_{(k-1)N})}\log(1+KN/d) \qquad (\det(A) \leqslant (Tr(A)/d)^d)
\end{aligned}
$$

which implies that,

$$
\frac{N\varepsilon}{d\log(1+KN/d)} \leqslant \frac{\det(\mathbf{A}_{kN})}{\det(\mathbf{A}_{(k-1)N})}
$$

Therefore,

$$
\begin{aligned}
|\mathcal{K}^+|\log\frac{N\varepsilon}{d\log(1+KN/d)} &\leqslant \sum_{k\in\mathcal{K}}\log\frac{\det(\mathbf{A}_{kN})}{\det(\mathbf{A}_{(k-1)N})} \leqslant \sum_{k=1}^{K}\log\frac{\det(\mathbf{A}_{kN})}{\det(\mathbf{A}_{(k-1)N})} \\
&= \log\frac{\det(\mathbf{A}_{KN})}{\det(\mathbf{A}_0)} \leqslant d\log(1+KN/d)
\end{aligned}
$$

which implies that, conditioning on $N \geqslant \frac{d}{\varepsilon}\log(1+KN/d)$, we have:

$$
|\mathcal{K}^+| \leqslant d\frac{\log(1+KN/d)}{\log(\frac{N\varepsilon}{d\log(1+KN/d)})}
$$

Now, we are interested in find the mimimum N, under the constraint that $|\mathcal{K}^+| \leqslant c_K d$. To solve this problem, we first choose an arbitrary $p \leqslant c_K$, and find a N such that,

$$
\frac{\log(1+KN/d)}{\log(\frac{N\varepsilon}{d\log(1+KN/d)})} \leqslant p
$$

In order to guarantee the above, we need:

$$
N\varepsilon \geqslant d\log(1+KN/d), \quad (\frac{N\varepsilon}{d\log(1+KN/d)})^p \geqslant 1+KN/d
$$

The first constraint can be satisfied easily with $N \geqslant c_1\frac{d}{\varepsilon}\log\frac{dH}{\varepsilon}$ for some constant c_1. Since usually $KN/d > 1$, the second constraint can be directly satisfied if:

$$
(\frac{N\varepsilon}{d\log(1+KN/d)})^p \geqslant 2KN/d
$$

Recall $K = c_K dH + 1$, it can be satisfied by choosing

$$
N \geqslant c_2\Big(c_K\frac{Hd^p}{\varepsilon^p}\log^p(\frac{Hd}{\varepsilon})\Big)^{\frac{1}{p-1}} \tag{7}
$$

where c_2 is an absolute constant. Therefore, we can find an absolute number c such that,

$$N = c\Big(c_K \frac{Hd^p}{\varepsilon^p}\log^p(\frac{Hd}{\varepsilon})\Big)^{\frac{1}{p-1}} \geqslant \max\{c_1\frac{d}{\varepsilon}\log(\frac{d}{\varepsilon}), c_2\Big(c_K \frac{Hd^p}{\varepsilon^p}\log^p(\frac{Hd}{\varepsilon})\Big)^{\frac{1}{p-1}}\}$$

to make sure that

$$|\mathcal{K}^+| \leqslant pd$$

Since in Eq.(7), it's required that $1/(p-1) < \infty$, we should constraint that $p > 1$ and therefore, $c_K \geqslant 2$. Because the dependence of $d, H, \frac{1}{\varepsilon}, \log\frac{dH}{\varepsilon}$ are decreasing as p increases, by assigning $p = c_K$ and $1 < p \leqslant c_K$, N will be minimized when $p = c_K$. Then, we finished the proof. □

C.3 Analysis for Algorithms

Next, we will use the above lemma to bound the difference between $J(\pi_K)$ and $J(\pi^*)$. We first prove a lemma similar to Lemma B.3 in (Jin et al., 2019) and Lemma A.1 in (Wang et al., 2020b).

Lemma C.4 (Concentration Lemma)**.** *We use $\mathcal{E}_1$ to denote the event that, when running Algorithm 1, the following inequality holds for all $k \in [K]$ and $h \in [h_k]$ and arbitrary V_{h+1}^k occurs in Alg 1.*

$$\Big\|\sum_{\tau=1}^{k-1}\sum_{n=1}^{N}\phi_h^{\tau n}\Big(V_{h+1}^k(s_{h+1}^{\tau n}) - \sum_{s'\in\mathcal{S}}P_h(s'|s_h^{\tau n}, a_h^{\tau n})V_{h+1}^k(s')\Big)\Big\|_{(\Lambda_h^k)^{-1}} \leqslant c\cdot dH\sqrt{\log(dKNH/\delta)}$$

Under Assumption A, there exists some absolute constant $c \geqslant 0$, such that $P(\mathcal{E}_1) \geqslant 1-\delta/2$.

Proof. The proof is almost identical to Lemma B.3 in (Jin et al., 2019), so we omit it here. The only difference is that we have an inner summation from $n = 1$ to N and we truncate the horizon at h_k in iteration k. □

Lemma C.5 (Overestimation)**.** *On the event $\mathcal{E}_1$ in Lemma C.4, which holds with probability $1-\delta/2$, for all $k \in [K]$ and $n \in [N]$,*

$$V_1^*(s_1^{kn}|h_k) \leqslant V_1^k(s_1^{kn})$$

where recall that V_1^k is the function computed at iteration k in Alg.1 and $V_1^*(\cdot|h_k) = \mathbb{E}[\sum_{h=1}^{h_k} r_h(s_h, a_h)|\pi^*_{[1:h_k]}]$ denote the optimal value function in the MDP truncated at layer h_k and $\pi^*_{[1:h_k]}$ is the optimal policy in the truncated MDP.

Besides, we also have:

$$\mathbb{E}_{s_1\sim d_1}[V_1^*(s_1|h_k) - V^{\pi_k}(s_1|h_k)] \leqslant 2\beta\mathbb{E}_{s_1,a_1,\ldots,s_{h_k},a_{h_k}\sim\pi_k}[\sum_{h=1}^{h_k}\|\phi(s_h,a_h)\|_{(\Lambda_h^k)^{-1}}]$$

Proof. First of all, by applying Lemma C.4 above, after a similar discussion to the proof of Lemma 3.1 in (Wang et al., 2020b), we can show that

$$|\phi(s,a)^\top w_h^k - \sum_{s'\in\mathcal{S}}P_h(s'|s,a)V_{h+1}^k(s')| \leqslant \beta\|\phi(s,a)\|_{(\Lambda_h^k)^{-1}}, \quad \forall s\in\mathcal{S}, a\in\mathcal{A}, h\in[h_k]$$

and the overestimation

$$V_h^*(s|h_k) \leqslant V_h^k(s), \quad \forall s\in\mathcal{S}, h\in[h_k]$$

As a result,

$$\begin{aligned}
&\mathbb{E}_{s_1\sim d_1}[V_1^*(s_1|h_k) - V^{\pi_k}(s_1|h_k)]\\
\leqslant&\mathbb{E}_{s_1\sim d_1}[V_1^k(s_1) - V^{\pi_k}(s_1|h_k)]\\
=&\mathbb{E}_{s_1\sim d_1, a_1\sim\pi_k}[Q_1^k(s_1,a_1) - Q^{\pi_k}(s_1,a_1|h_k)]\\
=&\mathbb{E}_{s_1\sim d_1, a_1\sim\pi_k}[\min\{(w_1^k)^\top\phi(s_1,a_1) + r_1(s_1,a_1) + u_1^k(s_1,a_1), H\} - r_1(s_1,a_1) - \sum_{s_2\in\mathcal{S}}P_1(s_2|s_1,a_1)V_2^{\pi_k}(s_2|h_k))]
\end{aligned}$$

$$
\begin{aligned}
&=\mathbb{E}_{s_1\sim d_1,a_1\sim\pi_k}[\min\{(w_1^k)^\top\phi(s_1,a_1),H-r_1(s_1,a_1)-u_1^k(s_1,a_1)\}-\sum_{s_2\in\mathcal{S}}P_1(s_2|s_1,a_1)V_2^k(s_2)]\\
&\quad+\mathbb{E}_{s_1\sim d_1,a_1\sim\pi_k}[\sum_{s_2\in\mathcal{S}}P_1(s_2|s_1,a_1)V_2^k(s_2)-\sum_{s_2\in\mathcal{S}}P_1(s_2|s_1,a_1)V_2^{\pi_k}(s_2|h_k))]+\mathbb{E}_{s\sim\mu}[u_1^k(s_1,a_1)]\\
&\leqslant\mathbb{E}_{s_1\sim d_1,a_1\sim\pi_k}[\sum_{s_2\in\mathcal{S}}P_1(s_2|s_1,a_1)V_2^k(s_2)-\sum_{s_2\in\mathcal{S}}P_1(s_2|s_1,a_1)V_2^{\pi_k}(s_2|h_k))]+2\mathbb{E}_{s\sim\mu}[u_1^k(s_1,a_1)]\\
&=\mathbb{E}_{s_1\sim d_1,a_1,s_2,a_2\sim\pi_k}[V_2^k(s_2)-V_2^{\pi_k}(s_2|h_k)]+2\mathbb{E}_{s\sim\mu}[u_1^k(s_1,a_1)]\\
&\leqslant\ldots\\
&\leqslant 2\mathbb{E}_{s_1\sim d_1,a_1,\ldots,s_{h_k},a_{h_k}\sim\pi_k}[\sum_{h=1}^{h_k}u_h^k(s_h,a_h)]\\
&\leqslant 2\beta\mathbb{E}_{s_1\sim d_1,a_1,\ldots,s_{h_k},a_{h_k}\sim\pi_k}[\sum_{h=1}^{h_k}\|\phi(s_h,a_h)\|_{(\Lambda_h^k)^{-1}}]
\end{aligned}
$$

where in the second inequality, we use the following fact

$$
\begin{aligned}
&\min\{(w_1^k)^\top\phi(s_1,a_1),H-r_1(s_1,a_1)-u_1^k(s_1,a_1)\}-\sum_{s_2\in\mathcal{S}}P_1(s_2|s_1,a_1)V_2^k(s_2)\\
&=\min\{(w_1^k)^\top\phi(s_1,a_1)-\sum_{s_2\in\mathcal{S}}P_1(s_2|s_1,a_1)V_2^k(s_2),H-r_1^k(s_1,a_1)-u_1^k(s_1,a_1)-\sum_{s_2\in\mathcal{S}}P_1(s_2|s_1,a_1)V_2^k(s_2)\}\\
&\leqslant\min\{\beta\|\phi(s_1,a_1)\|_{(\Lambda_1^k)^{-1}},H\}=u_1^k(s_1,a_1)
\end{aligned}
$$

□

Now we are ready to prove the following theorem restated from Theorem 4.1 in a more detailed version, where we include the guarantees during the execution of the algorithm.

Theorem C.6. *[Deployment Complexity] For arbitrary $\varepsilon,\delta>0$, and arbitrary $c_K\geqslant 2$, as long as $N\geqslant c\Big(c_K\frac{H^{4c_K+1}d^{3c_K}}{\varepsilon^{2c_K}}\log^{2c_K}(\frac{Hd}{\delta\varepsilon})\Big)^{\frac{1}{c_K-1}}$, where c is an absolute constant and independent with $c_K,d,H,\varepsilon,\delta$, by choosing*

$$K=c_KdH+1. \tag{8}$$

Algorithm 1 will terminate at iteration $k_H\leqslant K$ and return us a policy π^{k_H}, and with probability $1-\delta$, (1) $\mathbb{E}_{s_1\sim d_1}[V_1^*(s_1)-V_1^{\pi^{k_H}}(s_1)]\leqslant\varepsilon$. (2) for each $h\in[H-1]$, there exists an iteration k_h, such that $h_{k_h}=h$ but $h_{k_h+1}=h+1$, and π_{k_h} is an ε-optimal policy for the MDP truncated at step h;

Proof. As stated in the theorem, we use k_h to denote the number of deployment after which the algorithm switch the exploration from layer h to layer $h+1$, i.e. $h_{k_h}=h$ and $h_{k_h+1}=h+1$. According to the definition and the algorithm, we must have $\Delta_{k_h}\leqslant\frac{\varepsilon h_{k_h}}{2H}$, and for arbitrary $k_{h-1}+1\leqslant k\leqslant k_h-1$, $\Delta_k\geqslant\frac{\varepsilon h_k}{2H}$ (if $k_{h-1}+1>k_h-1$, then it means $\Delta_{k_{h-1}+1}$ is small enough and the algorithm directly switch the exploration to the next layer, and we can skip the discussion below). Therefore, for arbitrary $k_{h-1}+1\leqslant k\leqslant k_h-1$, during the k-th deployment, there exists $h\in[h_k]$, such that,

$$\frac{\varepsilon}{2H}\leqslant\frac{\Delta_k}{h_k}\leqslant\frac{2\beta}{N}\sum_{n=1}^N\|\phi(s_h^{kn},a_h^{kn})\|_{(\Lambda_h^k)^{-1}}\leqslant 2\beta\sqrt{\frac{1}{N}\sum_{n=1}^N\|\phi(s_h^{kn},a_h^{kn})\|^2_{(\Lambda_h^k)^{-1}}}$$

where the second inequality is because the average is less than the maximum. The above implies that

$$\frac{1}{N}\sum_{n=1}^N\|\phi(s_h^{kn},a_h^{kn})\|^2_{(\Lambda_h^k)^{-1}}=\frac{1}{N}Tr\Big((\Lambda_h^k)^{-1}\Big(\sum_{n=1}^N\phi(s_h^{kn},a_h^{kn})\phi(s_h^{kn},a_h^{kn})^\top\Big)\Big)\geqslant\frac{\varepsilon^2}{16H^2\beta^2} \tag{9}$$

According to Lemma 4.2, there exists constant c, c', such that by choosing N according to Eq.(10) below, the event in Eq.(9) will not happen more than dc_K times at each layer $h \in [h_k]$.

$$N \geqslant c\Big(c_K \frac{H^{4c_K+1}d^{3c_K}}{\varepsilon^{2c_K}} \log^{2c_K}(\frac{Hd}{\varepsilon\delta})\Big)^{\frac{1}{c_K-1}} \geqslant c'\Big(c_K \frac{H^{2c_K+1}d^{c_K}\beta^{2c_K}}{\varepsilon^{2c_K}} \log^{c_K}(\frac{Hd\beta}{\varepsilon})\Big)^{\frac{1}{c_K-1}} \tag{10}$$

Recall that $\varepsilon < 1$ and the covariance matrices in each layer is initialized by $I_{d\times d}$. Therefore, at the first deployment, although the computation of π^1 does not consider the layers $h \geqslant 2$, Eq.(9) happens in each layer $h \in [H]$. We use $\zeta(k, j)$ to denote the total number of times events in Eq.(9) happens for layer j previous to deployment k, as a result,

$$k_h \leqslant \sum_{j=1}^{h} \zeta(k_h, j) - (h-1) + h \leqslant c_K dh + 1, \quad \forall h \in [H]$$

where we minus $h-1$ because such event must happen at the first deployment for each $h \in [H]$ and we should remove the repeated computation; and we add another h back is because there are h times we waste the samples (i.e. for those k such that $\Delta_k < \frac{\varepsilon h_k}{2H}$). Therefore, we must have $k_H \leqslant c_K dH + 1 = K$.

Moreover, because at iteration $k = k_h$, we have $\Delta_{k_h} \leqslant \varepsilon/2$, according to Hoeffding inequality, with probability $1-\delta/2$, for each deployment k, we must have:

$$\mathbb{E}_{s_1,a_1,\ldots,s_{h_k},a_{h_k}\sim\pi_k}[2\beta \sum_{h=1}^{h_k} \|\phi(s_h,a_h)\|_{(\Lambda_h^k)^{-1}}] \leqslant \Delta_k + 2\beta H\sqrt{\frac{1}{2N}\log(\frac{K}{\delta})} \tag{11}$$

Therefore, by choosing

$$N \geqslant \frac{8\beta^2 H^2}{\varepsilon^2}\log(\frac{K}{\delta}) = O(\frac{d^2H^4}{\varepsilon^2}\log^2(\frac{K}{\delta})) \tag{12}$$

we must have,

$$\mathbb{E}_{s_1,a_1,\ldots,s_h,a_h\sim\pi_{k_h}}[2\beta \sum_{h'=1}^{h} \|\phi(s_{h'},a_{h'})\|_{(\Lambda_{h'}^k)^{-1}}] \leqslant \Delta_{k_h} + \frac{\varepsilon}{2} = \varepsilon, \quad \forall h \in [H]$$

Therefore, after a combination of Eq.(10) and Eq.(12), we can conclude that, for arbitrary $c_K \geqslant 2$, there exists absolute constant c, such that by choosing

$$N \geqslant c\Big(c_K \frac{H^{4c_K+1}d^{3c_K}}{\varepsilon^{2c_K}} \log^{2c_K}(\frac{Hd}{\varepsilon\delta})\Big)^{\frac{1}{c_K-1}}$$

the algorithm will stop at $k_H \leqslant K$, and with probability $1-\delta$ (on the event of $\mathcal{E}_1$ in C.4 and the Hoeffding inequality above), we must have:

$$\mathbb{E}_{s_1\sim d_1}[V_1^*(s_1) - V^{\pi_k}(s_1)] \leqslant \mathbb{E}_{s_1,a_1,\ldots,s_{h_k},a_{h_k}\sim\pi_{k+1}}[2\beta \sum_{h=1}^{h_k} \|\phi(s_h,a_h)\|_{(\Lambda_h^k)^{-1}}] \leqslant \varepsilon$$

and an additional benefits that for each $h \in [H-1]$, π_{k_h} is an ε-optimal policy at the MDP truncated at h step, or equivalently,

$$\mathbb{E}_{s_1\sim d_1}[V_1^*(s_1|h) - V_1^{\pi_{k_h}}(s_1|h)] \leqslant \varepsilon. \tag{13}$$

□

C.4 Additional Safety Guarantee Brought with Layer-by-Layer Strategy

The layer-by-layer strategy brings another advantage that, if we finish the exploration of the first h layers, based on the samples collected so far, we can obtain a policy $\widehat{\pi}_{|h}$, which is an ε-optimal in the MDP truncated at step h, or equivalently:

$$J(\pi^*) - J(\widehat{\pi}_{|h}) \geqslant H - h + O(\varepsilon), \quad \forall h \in [H]$$

We formally state these guarantees in Theorem C.6 (a detailed version of Theorem 4.1), Theorem D.4 and Theorem E.9 (the formal version of Theorem 4.4). Such a property may be valuable in certain application scenarios. For example, in "Safe DE-RL", which we will discuss in Appendix F, $\widehat{\pi}_{|h}$ can be used as the pessimistic policy in Algorithm 7 and guarantee the monotonic policy improvement criterion. Besides, in some real-world settings, we may hope to maintain a sub-optimal but gradually improving policy before we complete the execution of the entire algorithm.

If we replace Line 7-8 in LSVI-UCB (Algorithm 1) in Jin et al. (2019) with Line 13-18 in our Algorithm 1, the similar analysis can be done based on Lemma 4.2, and the same $\Theta(dH)$ deployment complexity can be derived. However, the direct extension based on LSVI-UCB does not have the above safety guarantee. It is only guaranteed to return a near-optimal policy after $K = \Theta(dH)$ deployments, but if we interrupt the algorithm after some $k < K$ deployments, there is no guarantee about what the best possible policy would be based on the data collected so far.

D Reward-Free Deployment-Efficient RL with Deterministic Policies

D.1 Algorithm

Similar to other algorithms in reward-free setting (Wang et al., 2020b; Jin et al., 2020), our algorithm includes an "Exploration Phase" to uniformly explore the entire MDP, and a "Planning Phase" to return near-optimal policy given an arbitrary reward function. The crucial part is to collect a well-covered dataset in the online "exploration phase", which is sufficient for the batch RL algorithm (Antos et al., 2008; Munos & Szepesvári, 2008; Chen & Jiang, 2019) in the offline "planning phase" to work.

Our algorithm in Alg.3 and Alg.4 is based on (Wang et al., 2020b) and the layer-by-layer strategy. The main difference with Algorithm 1 is in two-folds. First, similar to (Wang et al., 2020b), we replace the reward function with $1/H$ of the bonus term. Secondly, we use a smaller threshold for Δ_k comparing with Algorithm 1.

D.2 Analysis for Alg 3 and Alg 4

We first show a lemma adapted from Lemma C.4 for Alg 3. Since the proof is similar, we omit it here.

Lemma D.1 (Concentration for DE-RL in Reward-Free Setting)**.** *We use $\mathcal{E}_2$ to denote the event that, when running Algorithm 3, the following inequality holds for all $k \in [K]$ and $h \in [h_k]$ and all $V = V^k_{h+1}$ occurs in Alg 3 or $V = V_h$ occurs in Alg 4:*

$$\Big\| \sum_{\tau=1}^{k-1} \sum_{n=1}^{N} \phi_h^{\tau n} \Big(V(s_{h+1}^{\tau n}) - \sum_{s' \in \mathcal{S}} P_h(s'|s_h^{\tau n}, a_h^{\tau n}) V(s') \Big) \Big\|_{(\Lambda_h^k)^{-1}} \leqslant c \cdot dH \sqrt{\log(dKNH/\delta)}$$

Under Assumption A, there exists some absolute constant $c \geqslant 0$, such that $P(\mathcal{E}_2) \geqslant 1 - \delta/2$.

Proof. The proof is almost identical to Lemma 3.1 in (Wang et al., 2020b), so we omit it here. The only difference is that we have an inner summation from $n = 1$ to N and we truncate the horizon at h_k in iteration k. □

Next, we prove a lemma simlar to Lemma C.5 based on Lemma D.1.

Lemma D.2 (Overestimation)**.** *On the event $\mathcal{E}_2$ in Lemma D.1, which holds with probability $1-\delta/2$, in Algorithm 3, for all $k \in [K]$ and $n \in [N]$,*

$$V_1^*(s_1^{kn}, r^k|h_k) \leqslant V_1^k(s_1^{kn})$$

and

$$\mathbb{E}_{s_1 \sim d_1}[V_1^*(s, r^k|h_k)] \leqslant \mathbb{E}_{s_1 \sim d_1}[V_1^k(s)] \leqslant (2H+1)\mathbb{E}_{s_1 \sim d_1}[V^{\pi_k}(s, r^k|h_k)]$$

Proof. We first prove the overestimation inequality.

Algorithm 3: Reward-Free DE-RL with Deterministic Policies in Linear MDPs: Exploration Phase

Input: Failure probability $\delta > 0$, and target accuracy $\varepsilon > 0$, $\beta \leftarrow c_\beta \cdot dH\sqrt{\log(dH\delta^{-1}\varepsilon^{-1})}$ for some $c_\beta > 0$, total number of deployments K, batch size N
Initialize $h_1 = 1$
$D_1 = \{\}, D_2 = \{\}, ..., D_H = \{\}$
for $k = 1, 2, ..., K$ **do**
  $Q^k_{h_k+1}(\cdot,\cdot) \leftarrow 0$ and $V^k_{h_k+1}(\cdot) = 0$
  for $h = h_k, h_k - 1, ..., 1$ **do**
    $\Lambda^k_h \leftarrow I + \sum_{\tau=1}^{k}\sum_{n=1}^{N} \phi^{\tau n}_h (\phi^{\tau n}_h)^\top$
    $u^k_h(\cdot,\cdot) \leftarrow \min\{\beta \cdot \sqrt{\phi(\cdot,\cdot)^\top (\Lambda^k_h)^{-1}\phi(\cdot,\cdot)}, H\}$
    Define the exploration-driven reward function $r^k_h(\cdot,\cdot) \leftarrow u^k_h(\cdot,\cdot)/H$
    $w^k_h \leftarrow (\Lambda^k_h)^{-1} \sum_{\tau=1}^{k-1}\sum_{n=1}^{N} \phi^{\tau n}_h \cdot V^k_{h+1}(s^{\tau n}_{h+1})$
    $Q^k_h(\cdot,\cdot) \leftarrow \min\{(w^k_h)^\top \phi(\cdot,\cdot) + r^k_h(\cdot,\cdot) + u^k_h(\cdot,\cdot), H\}$ and $V^k_h(\cdot) = \max_{a\in\mathcal{A}} Q^k_h(\cdot,a)$
    $\pi^k_h(\cdot) \leftarrow \arg\max_{a\in\mathcal{A}} Q^k_h(\cdot,a)$
  end
  Define $\pi^k = \pi^k_1 \circ \pi^k_2 \circ ...\pi^k_{h_k} \circ \mathrm{unif}_{[h_k+1:H]}$
  for $n = 1, ..., N$ **do**
    Receive initial state $s^{kn}_1 \sim d_1$
    for $h = 1, 2, ..., H$ **do**
      Take action $a^{kn}_h \leftarrow \pi^k(s^{kn}_h)$ and observe $s^{kn}_{h+1} \sim P_h(s^k_h, a^k_h)$
      $D_h = D_h \bigcup \{(s^{kn}_h, a^{kn}_h)\}$
    end
  end
  Compute $\Delta_k \leftarrow \frac{2\beta}{N}\sum_{n=1}^{N}\sum_{h=1}^{h_k}\sqrt{\phi(s^{kn}_h, a^{kn}_h)^\top (\Lambda^k_h)^{-1}\phi(s^{kn}_h, a^{kn}_h)}$.
  if $\Delta_k < \frac{\varepsilon h_k}{(4H+2)H}$ **then**
    if $h_k = H$ **then return** $D = \{D_1, D_2, ..., D_H\}$;
    else $h_k \leftarrow h_k + 1$;
  end
end

Algorithm 4: Reward-Free DE-RL with Deterministic Policies in Linear MDPs: Planning Phase

Input: Horizon length $\widetilde{h}$; Dataset $\mathcal{D} = \{(s^{kn}_h, a^{kn}_h)_{k,n,h\in[K]\times[N]\times[\widetilde{h}]}\}$, reward function $r = \{r_h\}_{h\in[\widetilde{h}]}$
$Q_{\widetilde{h}+1}(\cdot,\cdot) \leftarrow 0$ and $V_{\widetilde{h}+1}(\cdot) \leftarrow 0$
for $h = \widetilde{h}, \widetilde{h}-1, ..., 1$ **do**
  $\Lambda_h \leftarrow I + \sum_{\tau=1}^{K}\sum_{n=1}^{N} \phi(s^{\tau n}_h, a^{\tau n}_h)\phi(s^{\tau n}_h, a^{\tau n}_h)^\top$
  Let $u^{plan}_h(\cdot,\cdot) \leftarrow \min\{\beta\sqrt{\phi(\cdot,\cdot)^\top (\Lambda_h)^{-1}\phi(\cdot,\cdot)}, \widetilde{h}\}$
  $w_h \leftarrow (\Lambda_h)^{-1}\sum_{\tau=1}^{K} \phi(s^{\tau n}_h, a^{\tau n}_h) \cdot V_{h+1}(s^{\tau n}_{h+1}, a)$
  $Q_h(\cdot,\cdot) \leftarrow \min\{w_h^\top \phi(\cdot,\cdot) + r_h(\cdot,\cdot) + u^{plan}_h(\cdot,\cdot), \widetilde{h}\}$ and $V_h(\cdot) = \max_{a\in\mathcal{A}} Q_h(\cdot,a)$
  $\pi_h(\cdot) \leftarrow \arg\max_{a\in\mathcal{A}} Q_h(\cdot,a)$
end
return $\pi_{r|\widetilde{h}} = \{\pi_h\}_{h\in[\widetilde{h}]}$, $\widehat{V}_1(\cdot, r|\widetilde{h}) := V_1(\cdot)$

Overestimation First of all, similar to the proof of Lemma 3.1 in (Wang et al., 2020b), on the event of $\mathcal{E}_2$ defined in Lemma C.4, which holds with probability $1-\delta/2$, we have:

$$|\phi(s,a)^\top w^k_h - \sum_{s'\in\mathcal{S}} P_h(s'|s,a)V^k_{h+1}(s')| \leqslant \beta \cdot \|\phi(s,a)\|_{(\Lambda^k_h)^{-1}}, \quad \forall s,a \in \mathcal{S}\times\mathcal{A},\ k\in[K],\ h\in[h_k] \tag{14}$$

Then, we can use induction to show the overestimation. For $h = h_k + 1$, we have:

$$0 = V^*_{h_k+1}(s, r^k|h_k) \leqslant V^k_{h_k+1}(s) = 0, \quad \forall s \in \mathcal{S}$$

Suppose for some $h \in [h_k]$, we have

$$V^*_{h+1}(s, r^k|h_k) \leqslant V^k_{h+1}(s), \quad \forall s \in \mathcal{S}$$

Then, $\forall s \in \mathcal{S}$, we have

$$\begin{aligned}
V^*_h(s, r^k|h_k) =& \max_a (r^k_h(s,a) + \sum_{s' \in \mathcal{S}} P_h(s'|s,a) V^*_{h+1}(s', r^k|h_k)) \\
\leqslant& \max_a (r^k_h(s,a) + \sum_{s' \in \mathcal{S}} P_h(s'|s,a) V^k_{h+1}(s')), H \\
\leqslant& \min\{\max_a (r^k_h(s,a) + \phi(s,a)^\top w^k_h + \beta \|\phi(s,a)\|_{(\Lambda^k_h)^{-1}}), H\} \\
=& \max_a \min\{r^k_h(s,a) + \phi(s,a)^\top w^k_h + \beta \|\phi(s,a)\|_{(\Lambda^k_h)^{-1}}, H\} \\
=& V^k_h(s)
\end{aligned}$$

where in the last inequality, we apply Eq.(14).

Relationship between $V^k_1(\cdot)$ and $V^{\pi_k}(\cdot, r^k)$

$$\begin{aligned}
& \mathbb{E}_{s_1 \sim d_1}[V^k_1(s_1) - V^{\pi_k}(s_1, r^k|h_k)] \\
=& \mathbb{E}_{s_1 \sim d_1, a_1 \sim \pi_k}[Q^k_1(s_1,a_1) - Q^{\pi_k}(s_1, a_1, r^k|h_k)] \\
=& \mathbb{E}_{s_1 \sim d_1, a_1 \sim \pi_k}[\min\{(w^k_1)^\top \phi(s_1,a_1) + r^k_1(s_1,a_1) + u^k_1(s_1,a_1), H\} \\
& - r^k_1(s_1,a_1) - \sum_{s_2 \in \mathcal{S}} P_1(s_2|s_1,a_1) V^{\pi_k}_2(s_2, r^k|h_k))] \\
\leqslant& \mathbb{E}_{s_1 \sim d_1, a_1 \sim \pi_k}[\min\{(w^k_1)^\top \phi(s_1,a_1), H - r^k_1(s_1,a_1) - u^k_1(s_1,a_1)\} - \sum_{s_2 \in \mathcal{S}} P_1(s_2|s_1,a_1) V^k_2(s_2)] \\
& + \mathbb{E}_{s_1 \sim d_1, a_1 \sim \pi_k}[\sum_{s_2 \in \mathcal{S}} P_1(s_2|s_1,a_1) V^k_2(s_2) - \sum_{s_2 \in \mathcal{S}} P_1(s_2|s_1,a_1) V^{\pi_k}_2(s_2, r^k|h_k))] + \mathbb{E}_{s \sim d_1}[u^k_1(s_1,a_1)] \\
\leqslant& \mathbb{E}_{s_1 \sim d_1, a_1 \sim \pi_k}[\sum_{s' \in \mathcal{S}} P_1(s_2|s_1,a_1) V^k_2(s_2) - \sum_{s_2 \in \mathcal{S}} P_1(s_2|s_1,a_1) V^{\pi_k}_2(s_2, r^k|h_k))] + 2\mathbb{E}_{s \sim d_1}[u^k_1(s_1,a_1)] \\
=& \mathbb{E}_{s_1 \sim d_1, a_1, s_2, a_2 \sim \pi_k}[V^k_2(s_2) - V^{\pi_k}_2(s_2, r^k|h_k)] + 2\mathbb{E}_{s \sim d_1}[u^k_1(s_1,a_1)] \\
\leqslant& \ldots \\
\leqslant& 2\mathbb{E}_{s_1 \sim d_1, a_1, \ldots, s_{h_k}, a_{h_k} \sim \pi_k}[\sum_{h=1}^{H} u^k_h(s_h, a_h)] \\
=& 2H\mathbb{E}_{s_1 \sim d_1}[V^{\pi_k}(s, r^k|h_k)]
\end{aligned}$$

where in the first inequality, we add and subtract $\sum_{s_2 \in \mathcal{S}} P_1(s_2|s_1,a_1) V^k_2(s_2)$, and in the second inequality, we use the following fact

$$\begin{aligned}
& \min\{(w^k_1)^\top \phi(s_1,a_1), H - r^k_1(s_1,a_1) - u^k_1(s_1,a_1)\} - \sum_{s_2 \in \mathcal{S}} P_1(s_2|s_1,a_1) V^k_2(s_2) \\
=& \min\{(w^k_1)^\top \phi(s_1,a_1) - \sum_{s_2 \in \mathcal{S}} P_1(s_2|s_1,a_1) V^k_2(s_2), H - r^k_1(s_1,a_1) - u^k_1(s_1,a_1) - \sum_{s_2 \in \mathcal{S}} P_1(s_2|s_1,a_1) V^k_2(s_2)\} \\
\leqslant& \min\{\beta \|\phi(s_1,a_1)\|_{(\Lambda^k_1)^{-1}}, H\} = u^k_1(s_1,a_1)
\end{aligned}$$

□

Next, we provide some analysis for Algorithm 4, which will help us to understand what we want to do in Algorithm 3

Lemma D.3. *On the event $\mathcal{E}_2$ in Lemma D.1, which holds with probability $1-\delta/2$, if we assign $\widetilde{h} = h_k$ in Algorithm 4 and assign $\mathcal{D}$ to be the samples collected till deployment k, i.e. $\mathcal{D} = \{(s_h^{kn}, a_h^{kn})_{k,n,h\in[K]\times[N]\times[h_k]}\}$, then for arbitrary reward function r satisfying the linear Assumption A, the policy $\pi_{r|\widetilde{h}}$ returned by Alg 4 would satisfy:*

$$\mathbb{E}_{s_1\sim d_1}[V_1^{\pi^*_{r|\widetilde{h}}}(s_1, r|\widetilde{h}) - V_1^{\pi_{r|\widetilde{h}}}(s_1, r|\widetilde{h})] \leqslant 2H\mathbb{E}_{s_1\sim d_1}[V_1^{\pi^*_{r^{plan}|\widetilde{h}}}(s_1, r^{plan}|\widetilde{h})] \tag{15}$$

where $r^{plan} := u^{plan}/\widetilde{h}$.

Proof. By applying the similar technique in the analysis of $\mathbb{E}_{s_1\sim d_1}[V_1^k(s_1) - V^{\pi_k}(s_1, r^k|\widetilde{h})]$ in Lemma D.2 after replacing r^k with r, we have:

$$\mathbb{E}_{s_1\sim d_1}[V_1^{\pi^*_{r|\widetilde{h}}}(s_1, r|\widetilde{h}) - V_1^{\pi_{r|\widetilde{h}}}(s_1, r|\widetilde{h})] \leqslant \mathbb{E}_{s_1\sim d_1}[\widehat{V}_1(s_1, r|\widetilde{h}) - V_1^{\pi_{r|\widetilde{h}}}(s_1, r|\widetilde{h})] \leqslant 2\mathbb{E}_{s_1\sim d_1}[V_1^{\pi_{r|\widetilde{h}}}(s_1, u^{plan})]$$

where $\widehat{V}_1$ denotes the value function returned by Alg 4 Besides,

$$2\mathbb{E}_{s_1\sim d_1}[V_1^{\pi_{r|\widetilde{h}}}(s_1, u^{plan})] = 2\widetilde{h}\mathbb{E}_{s_1\sim d_1}[V_1^{\pi_{r|\widetilde{h}}}(s_1, r^{plan}|\widetilde{h})] \leqslant 2H\mathbb{E}_{s_1\sim d_1}[V_1^{\pi^*_{r^{plan}|\widetilde{h}}}(s_1, r^{plan}|\widetilde{h})]$$

then, we finish the proof. □

From Eq.(15) in Lemma D.3, we can see that, after exploring with Algorithm 3, the sub-optimality gap between π^* and π returned by Alg.4 can be bounded by the value of the optimal policy w.r.t. r^K, which we will further bound in the next theorem.

Now we are ready to prove the main theorem.

Theorem D.4. *For arbitrary $\varepsilon, \delta > 0$, by assigning $K = c_K dH + 1$ for some $c_K \geqslant 2$, as long as*

$$N \geqslant c\Big(c_K \frac{H^{6c_K+1}d^{3c_K}}{\varepsilon^{2c_K}} \log^{2c_K}(\frac{Hd}{\delta\varepsilon})\Big)^{\frac{1}{c_K-1}}\Big\} \tag{16}$$

where c is an absolute constant and independent with $c_K, d, H, \varepsilon, \delta$, then, Alg 3 will terminate at iteration $k_H \leqslant K$ and return us a dataset $D = \{D_1, D_2, ..., D_H\}$, such that given arbitrary reward function r satisfying Assumption A, by running Alg 4 with D and r, with probability $1-\delta$, we can obtain a policy π_r satisfying $\mathbb{E}_{s\sim d_1}[V_1^*(s, r) - V^\pi(s, r)] \leqslant \varepsilon$.

Moreover, for each $h \in [H-1]$, there exists iteration k_h, such that $h_{k_h} = h$ but $h_{k_h+1} = h+1$, and if we run Alg 4 with the reward function r and the dataset Alg 3 has collected till $k = k_h$, we can obtain a policy $\pi_{r|h}$, which is an ε-optimal policy for MDP truncated at step h.

Proof. The proof is similar to Theorem 4.1. As stated in theorem, we use k_h to denote the number of deployment when the algorithm switch the exploration from layer h to layer $h+1$, i.e. $h_{k_h} = h$ and $h_{k_h+1} = h+1$. According to the definition and the algorithm, we must have $\Delta_{k_h} \leqslant \frac{\varepsilon h_{k_h}}{(4H+2)H}$, and for arbitrary $k_{h-1}+1 \leqslant k \leqslant k_h - 1$, we must have $\Delta_k \geqslant \frac{\varepsilon h_k}{(4H+2)H}$ (if $k_{h-1}+1 > k_h - 1$, then it means $\Delta_{k_{h-1}+1}$ is small enough and the algorithm directly switch the exploration to the next layer, and we can skip the discussion below). Therefore, for arbitrary $k_{h-1}+1 \leqslant k \leqslant k_h - 1$, during the k-th deployment, there exists $h \in [h_k]$, such that,

$$\frac{\varepsilon}{(4H+2)H} \leqslant \frac{\Delta_k}{h_k} \leqslant \frac{2\beta}{N}\sum_{n=1}^{N}\|\phi(s_h^{kn}, a_h^{kn})\|_{(\Lambda_h^k)^{-1}} \leqslant 2\beta\sqrt{\frac{1}{N}\sum_{n=1}^{N}\|\phi(s_h^{kn}, a_h^{kn})\|^2_{(\Lambda_h^k)^{-1}}}$$

which implies that

$$\frac{1}{N}\sum_{n=1}^{N}\|\phi(s_h^{kn}, a_h^{kn})\|^2_{(\Lambda_h^k)^{-1}} \geqslant \frac{\varepsilon^2}{16H^2(2H+1)^2\beta^2} \tag{17}$$

According to Lemma 4.2, there exists an absolute constant c, for arbitrary $\varepsilon < 1$, by choosing N according to Eq.(18) below, the events in Eq.(17) will not happen more than $c_K d$ times at each layer $h \in [H]$.

$$N \geqslant c\Big(c_K \frac{H^{6c_K+1} d^{3c_K}}{\varepsilon^{2c_K}} \log^{2c_K}(\frac{Hd}{\varepsilon\delta})\Big)^{\frac{1}{c_K-1}} \tag{18}$$

We use $\zeta(k,j)$ to denote the total number of times Eq.(9) happens for layer j till deployment k_h. With a similar discussion as Theorem 4.1, we have:

$$k_h \leqslant \sum_{j=1}^{h} \zeta(k_h, j) - (h-1) + h \leqslant c_K dh + 1, \quad \forall h \in [H]$$

Moreover, we must have $\Delta_{k_h} \leqslant \frac{\varepsilon}{4H+2}$ for each $h \in [H]$, and according to Hoeffding inequality, with probability $1-\delta/2$, for each step k, we must have

$$\mathbb{E}_{s_1,a_1,\ldots,s_h,a_h\sim\pi_{k_h}}[2\beta \sum_{h'=1}^{h} \|\phi(s_{h'},a_{h'})\|_{(\Lambda_{h'}^k)^{-1}}] \leqslant \Delta_{k_h} + 2\beta H\sqrt{\frac{1}{2N}\log(\frac{K}{\delta})}$$

Therefore, by choosing

$$N \geqslant \frac{8\beta^2 H^2 (2H+1)^2}{\varepsilon^2} \log(\frac{K}{\delta}) = O(\frac{d^2 H^6}{\varepsilon^2} \log^2(\frac{K}{\delta})) \tag{19}$$

we have,

$$\mathbb{E}_{s_1,a_1,\ldots,s_h,a_h\sim\pi_{k_h}}[2\beta \sum_{h'=1}^{h} \|\phi(s_{h'},a_{h'})\|_{(\Lambda_{h'}^k)^{-1}}] \leqslant \Delta_{k_h} + \frac{\varepsilon}{4H+2} = \frac{\varepsilon}{2H+1}$$

For arbitrary $h \in [H]$, in Algorithm 4, if we assign $\widetilde{h} = h$ and $\mathcal{D} = \{(s_h^{kn}, a_h^{kn})_{k,n,h\in[k_h]\times[N]\times[h]}\}$, note that $r^{plan} = r^{k_h}$, by applying Lemma D.2 and Lemma D.3 we have:

$$\begin{aligned}
&\mathbb{E}_{s_1\sim d_1}[V_1^{\pi^*_{r|\tilde{h}}}(s_1, r|\widetilde{h}) - V_1^{\pi_{r|\tilde{h}}}(s_1, r|\widetilde{h})] \leqslant 2H\mathbb{E}_{s_1\sim d_1}[V_1^{\pi^*_{r^{plan}|\tilde{h}}}(s_1, r^{plan}|\widetilde{h})] \\
=&2H\mathbb{E}_{s_1\sim d_1}[V_1^*(s_1, r^{k_h}|h)] \leqslant 2H(2H+1)\mathbb{E}_{s_1\sim d_1}[V^{\pi_{k_h}}(s_1, r^{k_h}|h)] \\
=&(2H+1)\mathbb{E}_{s_1,a_1,\ldots,s_h,a_h\sim\pi_{k_h}}[2\beta \sum_{h'=1}^{h} \|\phi(s_{h'},a_{h'})\|_{(\Lambda_{h'}^k)^{-1}}] \leqslant \varepsilon
\end{aligned}$$

Therefore, after a combination of Eq.(18) and Eq.(19), we can conclude that, for arbitrary $c_K \geqslant 2$, there exists absolute constant c, such that by choosing

$$N \geqslant c\Big(c_K \frac{H^{6c_K+1} d^{3c_K}}{\varepsilon^{2c_K}} \log^{2c_K}(\frac{Hd}{\delta\varepsilon})\Big)^{\frac{1}{c_K-1}}$$

Alg 3 will terminate at $k_H \leqslant K$, and with probability $1-\delta$ (on the event in Lemma D.1 and Hoeffding inequality above), for each $h \in [H]$, if we feed Alg 4 with $\widetilde{h} = h$, $\mathcal{D} = \{(s_h^{kn}, a_h^{kn})_{k,n,h\in[k_h]\times[N]\times[h]}\}$ and arbitrary linear reward function r, the policy $\pi_{r|h}$ returned by Alg 4 should satisfy:

$$\mathbb{E}_{s_1\sim d_1}[V_1^{\pi_{r|h}}(s_1, r|h)] \geqslant \mathbb{E}_{s_1\sim d_1}[V_1^{\pi^*_{r|h}}(s_1, r|h)] - \varepsilon$$

□

E DE-RL WITH ARBITRARY DEPLOYED POLICIES

In the proof for this section, without loss of generality, we assume the initial state is fixed, which will makes the notation and derivation simpler without trivialize the results. For the case where initial state is sampled from some fixed distribution, our algorithms and results can be extended simply by considering the concentration error related to the initial state distribution.

Algorithm 5: SolveOptQ

Input: Time step h; Dataset in previous steps $\{D_1, ..., D_{h-1}\}$; Unregularized Covariance Matrices $\{\Sigma_1, ...\Sigma_{h-1}\}$; Bonus factor β; Matrix to construct reward function Σ_R; Discretize resolution $\varepsilon_0 \leqslant \frac{1}{2d(N+1)}$

$R(\cdot,\cdot) \leftarrow \sqrt{\phi(\cdot,\cdot)^\top (2I + \Sigma_R)^{-1} \phi(\cdot,\cdot)}$

$Z_h \leftarrow \text{Discretization}((2I + \Sigma_R)^{-1}, \frac{{\varepsilon_0}^2}{4d}),\ \bar{R}(\cdot,\cdot) \leftarrow \sqrt{\phi(\cdot,\cdot)^\top Z_h \phi(\cdot,\cdot)}$

$Q_h(\cdot,\cdot) = R(\cdot,\cdot), \quad V_h(\cdot) = \max_a Q_h(\cdot,a), \bar{Q}_h(\cdot,\cdot) = \bar{R}(\cdot,\cdot), \quad \bar{V}_h(\cdot) = \max_a \bar{Q}_h(\cdot,a)$

$\bar{\pi}_h(\cdot) = \arg\max_a \bar{Q}_h(\cdot,a)$

for $\widetilde{h} = h-1, ..., 1$ **do**

  $w_{\widetilde{h}} \leftarrow \Sigma_{\widetilde{h}}^{-1} \sum_{(s_{\widetilde{h}}, a_{\widetilde{h}}, s_{\widetilde{h}+1}) \in D_{\widetilde{h}}} \phi(s_{\widetilde{h}}, a_{\widetilde{h}}) \cdot V_{\widetilde{h}+1}(s_{\widetilde{h}+1})$

  $u_{\widetilde{h}} := \beta[\phi(\cdot,\cdot)^\top \Sigma_{\widetilde{h}}^{-1} \phi(\cdot,\cdot)]^{1/2}$

  $Q_{\widetilde{h}}(\cdot,\cdot) \leftarrow \min\{w_{\widetilde{h}}^\top \phi(\cdot,\cdot) + u_{\widetilde{h}}, 1\}, V_{\widetilde{h}}(\cdot) \leftarrow \max_a Q_{\widetilde{h}}(\cdot,a)$

  $\bar{w}_{\widetilde{h}} \leftarrow \text{Discretization}(w_{\widetilde{h}}, \frac{\varepsilon_0}{2d}), Z_{\widetilde{h}} \leftarrow \text{Discretization}(\beta^2 \Sigma_{\widetilde{h}}^{-1}, \frac{{\varepsilon_0}^2}{4d})$

  $\bar{u}_{\widetilde{h}} := [\phi(\cdot,\cdot)^\top Z_{\widetilde{h}} \phi(\cdot,\cdot)]^{1/2}$

  $\bar{Q}_{\widetilde{h}}(\cdot,\cdot) \leftarrow \min\{\bar{w}_{\widetilde{h}}^\top \phi(\cdot,\cdot) + \bar{u}_{\widetilde{h}}, 1\}, \bar{V}_{\widetilde{h}}(\cdot) \leftarrow \max_a \bar{Q}_{\widetilde{h}}(\cdot,a)$

  $\bar{\pi}_{\widetilde{h}}(\cdot) \leftarrow \arg\max_a \bar{Q}_{\widetilde{h}}(\cdot,a)$

end

return $V_1(s_1), \bar{\pi} := \bar{\pi}_1 \circ \bar{\pi}_2 \circ ... \bar{\pi}_h$

Algorithm 6: EstimateCovMatrix

Input: Time step h; Dataset in previous steps $\{D_1, ..., D_{h-1}\}$; Covariance Matrices $\{\Sigma_1, ...\Sigma_{h-1}\}$; Deterministic Policy to evaluate $\bar{\pi} = \{\bar{\pi}_1, \bar{\pi}_2, ..., \bar{\pi}_h\}$

Initialize a zero matrix $\widetilde{\Lambda}_h^{\bar{\pi}} = O$

for $i = 1, 2, ..., d$ **do**

  for $j = i, i+1, ..., d$ **do**

    Define $\widetilde{R}^{ij}$, such that, $\widetilde{R}_h^{ij}(\cdot,\cdot) = \frac{1+\phi_i(\cdot,\cdot)\phi_j(\cdot,\cdot)}{2}$ and $\widetilde{R}_{\widetilde{h}}^{ij} = 0$ for all $\widetilde{h} \in [h-1]$;

    $\widehat{Q}_h^{\bar{\pi}}(\cdot,\cdot) = \widetilde{R}_h^{ij}(\cdot,\cdot), \widehat{V}_h^{\bar{\pi}}(\cdot) = \widehat{Q}_h^{\bar{\pi}}(\cdot, \bar{\pi}_h(\cdot))$

    for $\widetilde{h} = h-1, ..., 1$ **do**

      $\widehat{w}_{\widetilde{h}}^{\bar{\pi}} \leftarrow \Sigma_{\widetilde{h}}^{-1} \sum_{(s_{\widetilde{h}}, a_{\widetilde{h}}, s_{\widetilde{h}+1}) \in D_{\widetilde{h}}} \phi(s_t, a_t) \cdot \widehat{V}_{\widetilde{h}+1}^{\bar{\pi}}(s_{\widetilde{h}+1})$

      $\widehat{Q}_{\widetilde{h}}^{\bar{\pi}}(\cdot,\cdot) \leftarrow \min\{\widetilde{R}_{\widetilde{h}}^{ij}(\cdot,\cdot) + (\widehat{w}_{\widetilde{h}}^{\bar{\pi}})^\top \phi(\cdot,\cdot), 1\}$

      $\widehat{V}_{\widetilde{h}}^{\bar{\pi}}(\cdot) = \widehat{Q}_{\widetilde{h}}^{\bar{\pi}}(\cdot, \bar{\pi}_{\widetilde{h}}(\cdot))$

    end

    $(\widetilde{\Lambda}_{\widetilde{h}}^{\bar{\pi}})_{ij} \leftarrow \widehat{V}_1^{\bar{\pi}}(s_1); \quad (\widetilde{\Lambda}_{\widetilde{h}}^{\bar{\pi}})_{ji} \leftarrow \widehat{V}_1^{\bar{\pi}}(s_1)$

  end

end

$\widehat{\Lambda}_{\widetilde{h}}^{\bar{\pi}} = 2(\widetilde{\Lambda}_{\widetilde{h}}^{\bar{\pi}}) - \mathbf{1}$

return $\widehat{\Lambda}_{\widetilde{h}}^{\bar{\pi}}$

E.1 ALGORITHMS

We first introduce the definition for Discretization function:

Definition E.1 (Discretization function)**.** Given vector $w = (w_1, w_2, ..., w_d)^\top \in \mathbb{R}^d$ or matrix $\Sigma = (\Sigma_{ij})_{i,j \in [d]} \in \mathbb{R}^{d \times d}$ as input, we have:

$$\text{Discretization}(w, \varepsilon_0) = (\varepsilon_0 \lceil \frac{w_1}{\varepsilon_0} \rceil, \varepsilon_0 \lceil \frac{w_2}{\varepsilon_0} \rceil, ..., \varepsilon_0 \lceil \frac{w_d}{\varepsilon_0} \rceil), \quad \text{Discretization}(\Sigma, \varepsilon_0) = (\varepsilon_0 \lceil \frac{\Sigma_{ij}}{\varepsilon_0} \rceil)_{i,j \in [d]}$$

where $\lceil \cdot \rceil$ is the ceiling function.

In Algorithm 6, we are trying to estimate the expected covariance matrix under policy $\bar{\pi}$ by policy evaluation. The basic idea is that, the expected covariance matrix can be represented by:

$$\mathbb{E}_{s_h,a_h\sim\bar{\pi}}[\phi(s_h,a_h)\phi(s_h,a_h)^\top] = \left(\mathbb{E}_{\bar{\pi}}[\phi_i(s_h,a_h)\phi_j(s_h,a_h)]\right)_{ij} = \left(V^{\bar{\pi}}(s_1,R^{ij})\right)_{ij}$$

where we use $(a_{ij})_{ij}$ to denote a matrix whose element indexed by i in row and j in column is a_{ij}. In another word, the element in the covariance matrix indexed by ij is equal to the value function of policy $\bar{\pi}$ with $R_h^{ij}(s_h,a_h) := \phi_i(s_h,a_h)\phi_j(s_h,a_h)$ as reward function at the last layer (and use zero reward in previous layers), where ϕ_i denotes the i-th elements of vector ϕ. Because the techniques rely on the reward is non-negative and bounded in $[0,1]$, by leveraging the fact that $|\phi_i(\cdot,\cdot)| \leqslant \|\phi(\cdot,\cdot)\| \leqslant 1$, we shift and scale R^{ij} to obtain $\widetilde{R}^{ij}$ and use it for policy evaluation.

In Alg 5, we maintance two Q functions Q_h and $\bar{Q}_h$. The learning of Q_h is based on LSVI-UCB, while $\bar{Q}_h$ is a "discretized version" for Q_h computed by discretizing $w_h, \beta^2\Sigma_h^{-1}$ (or Σ_R^{-1} at layer h) elementwisely with resolution ε_0, and $\bar{Q}_h$ will be used to compute $\bar{\pi}_h$ for deployment. The main reason why we discretize Q_h is to make sure the number of greedy policies $\bar{\pi}$ is bounded, so that we can use union bound and upper bound the error when using Alg6 to estimate the covariance matrix. In Section E.5, we will analyze the error resulting from discretization, and we will upper bound the estimation error Algorithm 6.

E.2 Function Classes and ε_0-Cover

We first introduce some useful function classes and their ε_0-cover.

Notation for Value Function Classes and Policy Classes We first introduce some new notations for value and policy classes. Similar to Eq.(6) in (Jin et al., 2019), we define the greedy value function class

$$\mathcal{V}^*_{L,B} = \{V(\cdot)|V(\cdot) = \max_a \min\{\phi(\cdot,a)^\top w + \sqrt{\phi(\cdot,a)^\top\Sigma\phi(\cdot,a)}, 1\}, \|w\| \leqslant L, \|\Sigma\| \leqslant B\}$$

and the Q function class:

$$\mathcal{Q}_{L,B} = \{Q(\cdot,\cdot)|Q(\cdot,\cdot) = \min\{\phi(\cdot,\cdot)^\top w + \sqrt{\phi(\cdot,\cdot)^\top\Sigma\phi(\cdot,\cdot)}, 1\}, \|w\| \leqslant L, \|\Sigma\| \leqslant B\}$$

Besides, suppose we have a deterministic policy class Π with finite candidates (i.e. $|\Pi| \leqslant \infty$), we use $\mathcal{V}_{L,B} \times \Pi$ to denote:

$$\mathcal{V}_{L,B} \times \Pi = \{V(\cdot)|V(\cdot) = \min\{\phi(\cdot,\pi(\cdot))^\top w + \sqrt{\phi(\cdot,\pi(\cdot))^\top\Sigma\phi(\cdot,\pi(\cdot))}, 1\}, \|w\| \leqslant L, \|\Sigma\| \leqslant B, \pi \in \Pi\}$$

Recall that in Alg.6, we will use a special reward function, and we need to consider it in the union bound. We denote:

$$\mathcal{V}_\phi \times \Pi = \{V|V(\cdot) = \frac{1+\phi_i(\cdot,\pi(\cdot))\phi_j(\cdot,\pi(\cdot))}{2}, i,j \in [d], \pi \in \Pi\}$$

and easy to check $|\mathcal{V}_\phi \times \Pi| = d^2|\Pi|$.

Moreover, if we have a Q function class $\mathcal{Q}$, we will use $\Pi_{\mathcal{Q}}$ to denote the class of greedy policies induced from $\mathcal{Q}$, i.e.

$$\Pi_{\mathcal{Q}} := \{\pi(\cdot) = \arg\max Q(\cdot,a)|Q \in \mathcal{Q}\}.$$

Discretization with Resolution ε_0 In the following, we will use $\mathcal{C}_{w,L,\varepsilon_0}$ to denote the ε_0-cover for $w \in \mathbb{R}^d$ with $\|w\| \leqslant L$, concretely,

$$\mathcal{C}_{w,L,\varepsilon_0} = \{w||\frac{w_i}{\varepsilon_0}| \in [\lceil\frac{L}{\varepsilon_0}\rceil], \forall i \in [d]\}$$

where $\lceil\cdot\rceil$ is the ceiling function.

Similarly, we will use $\mathcal{C}_{\Sigma,B,\varepsilon_0}$ to denote the ε_0-cover for matrix $\Sigma \in \mathbb{R}^{d\times d}$ with $\max_{i,j}|\Sigma_{ij}| \leqslant B$

$$\mathcal{C}_{\Sigma,B,\varepsilon_0} = \{\Sigma||\frac{\Sigma_{ij}}{\varepsilon_0}| \in [\lceil\frac{B}{\varepsilon_0}\rceil], \forall i,j \in [d]\}.$$

Easy to check that:

$$\log|\mathcal{C}_{w,L,\varepsilon_0}| \leqslant d\log\frac{2L}{\varepsilon_0}, \quad \log|\mathcal{C}_{\Sigma,B,\varepsilon_0}| \leqslant d^2\log\frac{2B}{\varepsilon_0}$$

Recall the definition of Discretize function in Def. E.1, easy to check that:

$$\|\text{Discretize}(w,\varepsilon_0)-w\| \leqslant d\varepsilon_0, \|\text{Discretize}(\Sigma,\varepsilon_0)-\Sigma\| \leqslant \|\text{Discretize}(\Sigma,\varepsilon_0)-\Sigma\|_F \leqslant d\varepsilon_0$$

ε_0-cover Before we introduce our notations for ε_0-net, we first show a useful lemma:

Lemma E.2. *For arbitrary w,Σ, denote $\bar{w} = \text{Discretize}(w,\frac{\varepsilon_0}{2d})$ and $\bar{\Sigma} = \text{Discretize}(\Sigma,\frac{{\varepsilon_0}^2}{4d})$. Consider the following two functions and their greedy policies, where $\|\phi(\cdot,\cdot)\| \leqslant 1$*

$$Q(s,a) = \min\{w^\top\phi(\cdot,a)+\sqrt{\phi(\cdot,a)^\top\Sigma\phi(\cdot,a)},1\}, \quad \pi = \arg\max_a Q(s,a)$$
$$\bar{Q}(s,a) = \min\{\bar{w}^\top\phi(\cdot,a)+\sqrt{\phi(\cdot,a)^\top\bar{\Sigma}\phi(\cdot,a)},1\}, \quad \bar{\pi} = \arg\max_a \bar{Q}(s,a)$$

then we have:

$$|Q(s,\pi(s))-Q(s,\bar{\pi}(s))| \leqslant 2\varepsilon_0, \quad \forall s\in\mathcal{S},$$
$$\|Q-\bar{Q}\|_\infty \leqslant \varepsilon_0, \quad \sup_s|\max_a Q(\cdot,a)-\max_a\bar{Q}(\cdot,a)| \leqslant \varepsilon_0$$

Proof. After similar derivation as Eq.(28) in (Jin et al., 2019), we can show that

$$\begin{aligned}\sup_s|\max_a Q(\cdot,a)-\max_a\bar{Q}(\cdot,a)| \leqslant& \|Q-\bar{Q}\|_\infty\\ \leqslant& \sup_{s,a}\left|w^\top\phi(\cdot,\cdot)+\sqrt{\phi(\cdot,\cdot)^\top\Sigma\phi(\cdot,\cdot)}-\bar{w}^\top\phi(\cdot,\cdot)-\sqrt{\phi(\cdot,\cdot)^\top\bar{\Sigma}\phi(\cdot,\cdot)}\right|\\ \leqslant& \|w-\bar{w}\|+\sqrt{\|\Sigma-\bar{\Sigma}\|_F} \leqslant d\frac{\varepsilon_0}{2d}+\sqrt{d\frac{{\varepsilon_0}^2}{4d}} \leqslant \varepsilon_0\end{aligned}$$

Because π and $\bar{\pi}$ are greedy policies, we have:

$$\begin{aligned}0 \leqslant& Q(s,\pi(s))-Q(s,\bar{\pi}(s)) \leqslant Q(s,\pi(s))-\bar{Q}(s,\pi(s))+\bar{Q}(s,\bar{\pi}(s))-Q(s,\bar{\pi}(s))\\ \leqslant& 2\|Q-\bar{Q}\|_\infty \leqslant 2\varepsilon_0.\end{aligned}$$

□

Now, we consider the following Q function class and V function class,

$$\bar{\mathcal{Q}}_{L,B,\varepsilon_0} := \{Q|Q(\cdot,\cdot) = \min\{w^\top\phi(\cdot,\cdot)+\sqrt{\phi(\cdot,\cdot)^\top\Sigma\phi(\cdot,\cdot)},1\}, w\in\mathcal{C}_{w,L,\frac{\varepsilon_0}{2d}}, \Sigma\in\mathcal{C}_{\Sigma,dB,\frac{{\varepsilon_0}^2}{4d}}\}$$
$$\bar{\mathcal{V}}^*_{L,B,\varepsilon_0} := \{V|V(\cdot) = \max_a\min\{w^\top\phi(\cdot,a)+\sqrt{\phi(\cdot,a)^\top\Sigma\phi(\cdot,a)},1\}, w\in\mathcal{C}_{w,L,\frac{\varepsilon_0}{2d}}, \Sigma\in\mathcal{C}_{\Sigma,dB,\frac{{\varepsilon_0}^2}{4d}}\}$$

based on Lemma E.2, and another important fact that $\max_{i,j}|a_{ij}| \leqslant \|A\|_F \leqslant d\|A\|$, we know that $\bar{\mathcal{Q}}_{L,B,\varepsilon_0}$ is an ε_0-cover of $\mathcal{Q}_{L,B}$, i.e. for arbitrary $Q\in\mathcal{Q}_{L,B}$, there exists $\bar{Q}\in\bar{\mathcal{Q}}_{L,B,\varepsilon_0}$, such that $\|Q-\bar{Q}\| \leqslant \varepsilon_0$. Similarly, $\bar{\mathcal{V}}^*_{L,B,\varepsilon_0}$ is also an ε_0-cover of $\mathcal{V}^*_{L,B}$.

Besides, we will use $\Pi_{\bar{\mathcal{Q}}_{L,B,\varepsilon_0}}$ to denote the collection of greedy policy induced from elements in $\bar{\mathcal{Q}}_{L,B,\varepsilon_0}$.

We also define $\bar{\mathcal{V}}_{L,B,\varepsilon_0}\times\Pi$, which is an ε_0 cover for $\mathcal{V}_{L,B}\times\Pi$.

$$\bar{\mathcal{V}}_{L,B,\varepsilon_0}\times\Pi := \{V|V(\cdot) = \min\{\phi(\cdot,\pi(\cdot))^\top w+\sqrt{\phi(\cdot,\pi(\cdot))^\top\Sigma\phi(\cdot,\pi(\cdot))},1\}, w\in\mathcal{C}_{w,L,\frac{\varepsilon_0}{2d}}, \Sigma\in\mathcal{C}_{\Sigma,dB,\frac{{\varepsilon_0}^2}{4d}}, \pi\in\Pi\}$$

Obviously, $|\bar{\mathcal{V}}_{L,B,\varepsilon_0}\times\Pi| = |\Pi|\cdot|\mathcal{C}_{w,L,\frac{\varepsilon_0}{2d}}|\cdot|\mathcal{C}_{\Sigma,dB,\frac{{\varepsilon_0}^2}{4d}}|$.

Besides, because $\mathcal{V}_\phi\times\Pi$ is already a finite set, it is an ε_0-cover of itself.

E.3 CONSTRAINTS IN ADVANCE

Induction Condition Related to Accumulative Error Recall the induction condition in 4.5, and we restate it here.

Condition 4.5. [Induction Condition] Suppose after $h-1$ deployments, we have the following induction condition for some $\xi < 1/d$, which will be determined later:

$$\max_\pi \mathbb{E}_\pi[\sum_{\widetilde{h}=1}^{h-1} \sqrt{\phi(s_{\widetilde{h}}, a_{\widetilde{h}})^\top \Sigma_{\widetilde{h}}^{-1} \phi(s_{\widetilde{h}}, a_{\widetilde{h}})}] \leqslant \tfrac{h-1}{H}\xi. \tag{2}$$

Constraints for the Validity of the Algorithm Besides, in order to make sure the algorithm can run successfully, we add the following constaints:

$$\Sigma_R \geq -\frac{1}{2}I \tag{20}$$

$$Z_{\widetilde{h}} \geq 0, \quad \forall \widetilde{h} \in [h-1] \tag{21}$$

$$I \geq Z_h \geq 0 \tag{22}$$

where constraint (22) for Z_h is to make sure the reward $\bar{R}$ locates in $[0,1]$ interval.

Recall the definition of $\Sigma_{\widetilde{h}} = I + \sum_{n=1}^N \phi(s_{\widetilde{h}}, a_{\widetilde{h}})\phi(s_{\widetilde{h}}, a_{\widetilde{h}})^\top$, therefore,

$$\sigma_{\min}(\beta^2 \Sigma_{\widetilde{h}}^{-1}) = \beta^2/\sigma_{\max}(\Sigma_{\widetilde{h}}) \geqslant \frac{\beta^2}{1+N}$$

According to Lemma E.11, to make sure $Z_{\widetilde{h}} \geq 0$, we need the following constraint on ε_0

$$d\frac{{\varepsilon_0}^2}{4d} \leqslant \frac{\beta^2}{(N+1)} \tag{23}$$

which is equivalent to $\varepsilon_0 \leqslant \beta/\sqrt{N+1}$.

As for Z_h, the constraint is equivalent to:

$$2I + \Sigma_R \geq (1 + \frac{{\varepsilon_0}^2}{4})I, \quad (2I + \Sigma_R)^{-1} \geq \frac{{\varepsilon_0}^2}{4} I$$

and can be rewritten to

$$I + \Sigma_R \geq \frac{{\varepsilon_0}^2}{4} I, \quad \frac{4}{{\varepsilon_0}^2} I \geq 2I + \Sigma_R \tag{24}$$

E.4 CONCENTRATION BOUND

Based on the notations above, we are already to claim that:

Claim 3. *By choosing $L = \sqrt{dN}, B = \beta^2 + d\varepsilon_0$ for some $\varepsilon_0 \leqslant 1/d$, during the running of Algorithm 2*

- *In Alg 5, for each $h \in [H]$ and $\widetilde{h} \in [h]$, and the $\bar{Q}_{\widetilde{h}}$ and $\bar{\pi}_{\widetilde{h}}$ generated while running the algorithm, we must have $\bar{Q}_{\widetilde{h}} \in \bar{Q}_{L,B,\varepsilon_0}$ and $\bar{\pi}_{\widetilde{h}} \in \Pi_{\bar{Q}_{L,B,\varepsilon_0}}$. Besides, $Q_{\widetilde{h}} \in \mathcal{Q}_{L,B}$ and $V_{\widetilde{h}} \in \mathcal{V}^*_{L,B}$*
- *In Alg 6, for all $\widetilde{h} \in [h]$, we have $\bar{\pi}_{\widetilde{h}} \in \Pi_{\bar{Q}_{L,B,\varepsilon_0}}$, and for arbitrary value function $\widehat{V}^{\bar{\pi}}_{\widetilde{h}}$ generated in it, we have $\widehat{V}^{\bar{\pi}}_{\widetilde{h}} \in (\mathcal{V}_{L,B} \times \Pi_{\bar{Q}_{L,B,\varepsilon_0}}) \cup (\mathcal{V}_\phi \times \Pi_{\bar{Q}_{L,B,\varepsilon_0}})$ and therefore, there exists $V \in (\bar{\mathcal{V}}_{L,B,\varepsilon_0} \times \Pi_{\bar{Q}_{L,B,\varepsilon_0}}) \cup (\mathcal{V}_\phi \times \Pi_{\bar{Q}_{L,B,\varepsilon_0}})$ such that $\|V - \widehat{V}^{\bar{\pi}}_{\widetilde{h}}\| \leqslant \varepsilon_0$*

Proof. **Algorithm 5**: First we bound the norm of the weights w_h in Algorithm 5. For arbitrary $v \in \mathbb{R}^d$ and $\|v\| = 1$, we have:

$$|v^\top w_{\widetilde{h}}| = |v\Sigma_{\widetilde{h}}^{-1} \sum_{(s_{\widetilde{h}}, a_{\widetilde{h}}, s_{\widetilde{h}+1}) \in D_{\widetilde{h}}} \phi(s_{\widetilde{h}}, a_{\widetilde{h}}) V_{\widetilde{h}+1}(s_{\widetilde{h}+1})| \leqslant |v\Sigma_{\widetilde{h}}^{-1} \sum_{(s_{\widetilde{h}}, a_{\widetilde{h}}, s_{\widetilde{h}+1}) \in D_{\widetilde{h}}} \phi(s_{\widetilde{h}}, a_{\widetilde{h}})|$$

$$\leqslant \sqrt{|\sum_{(s_{\tilde{h}},a_{\tilde{h}},s_{\tilde{h}+1})\in D_{\tilde{h}}} v^\top \Sigma_{\tilde{h}}^{-1} v| \sum_{(s_{\tilde{h}},a_{\tilde{h}},s_{\tilde{h}+1})\in D_{\tilde{h}}} \phi(s_{\tilde{h}},a_{\tilde{h}})^\top \Sigma_{\tilde{h}}^{-1} \phi(s_{\tilde{h}},a_{\tilde{h}})|}$$
$$\leqslant \|v\| \sqrt{d|D_{\tilde{h}}|} = \|v\|\sqrt{dN}$$

therefore, $\|w_{\tilde{h}}\| \leqslant \sqrt{dN}$. Besides, according to Lemma E.11 and constraint (24), we have:

$$\|\beta^2 \Sigma_{\tilde{h}}^{-1}\| \leqslant \beta^2, \quad \|Z_{\tilde{h}}\| \leqslant \|\beta^2 \Sigma_{\tilde{h}}^{-1}\| + d\varepsilon_0 \leqslant \beta^2 + d\varepsilon_0 \quad \forall h \in [h-1]$$
$$\|(2I + \Sigma_R)^{-1}\| \leqslant 1, \quad \|Z_h\| \leqslant \|(2I+\Sigma_R)^{-1}\| + d\varepsilon_0 \leqslant 1 + d\varepsilon_0$$

Recall $B = \beta^2 + d\varepsilon_0$ and $\beta > 1$, the claim about Alg 5 is true.

Algorithm 6: The discussion about the value range of $\widehat{w}_{\tilde{h}}^{\bar{\pi}}$ is similar to above. Therefore, all the value functions occurred in the previous $h-1$ layers would belong to $\mathcal{V}_{L,B} \times \Pi$, except that the last layer should belong to $\mathcal{V}_\phi \times \Pi$. Besides, since Alg 6 is only used to estimate the policies returned by Alg 5, we should have $\Pi = \Pi_{\bar{\mathcal{Q}}_{L,B,\varepsilon_0}}$. As a result, the claim for Alg 6 is correct. □

Recall Lemma D.4 from (Jin et al., 2019), which holds for arbitrary $\mathcal{V}$ with covering number $\mathcal{N}_{\varepsilon_0}$ and $\sup_s |V(s)| \leqslant H$. Next, we state a slightly generalized version by replacing $\sup_s |V(s)| \leqslant H$ with $\sup_s |V(s)| \leqslant 1$, since in our Alg. 5 and Alg. 6 $V_{\max} = 1$. This is also the main reason why we only need the coefficient of the bonus term $\beta = \widetilde{O}(d)$ instead of $\widetilde{O}(dH)$ in Alg. 5 so that we can achieve better dependence on H.

Lemma E.3 ((Revised) Lemma D.4 in (Jin et al., 2019))**.** *Let $\{s_\tau\}_{\tau=1}^\infty$, be a stochastic process on state space $\mathcal{S}$ with corresponding filtration $\{\mathcal{F}_\tau\}_{\tau=1}^\infty$. Let $\{\phi_\tau\}_{\tau=0}^\infty$ be an $\mathbb{R}^d$-valued stochastic process where $\phi_\tau \in \mathcal{F}_{\tau-1}$ and $\|\phi_\tau\| \leqslant 1$. Let $\Lambda_t = \lambda I + \sum_{\tau=1}^t \phi_\tau \phi_\tau^\top$. Then for any $\delta > 0$, with probability at least $1-\delta$, for all $t \geqslant 0$, and any $V \in \mathcal{V}$ so that $\sup_s |V(s)| \leqslant 1$, we have:*

$$\Big\| \sum_{\tau=1}^t \phi_\tau \{V(s_\tau) - \mathbb{E}[V(x_\tau)|\mathcal{F}_{\tau-1}]\} \Big\|_{\Lambda_t^{-1}}^2 \leqslant 4[\frac{d}{2}\log\frac{t+\lambda}{\lambda} + \log\frac{\mathcal{N}_{\varepsilon_0}}{\delta}] + \frac{8t^2{\varepsilon_0}^2}{\lambda}$$

where $\mathcal{N}_{\varepsilon_0}$ is the ε_0-covering number of $\mathcal{V}$ w.r.t. the distance $\mathrm{dist}(V,V') = \sup_s |V(s) - V(s)'|$

Now we are ready to prove the main concentration result for Algorithm 2:

Theorem E.4. *Consider value function class $\mathcal{V} := \mathcal{V}_{L,B}^* \cup (\mathcal{V}_{L,B} \times \Pi_{\bar{\mathcal{Q}}_{L,B,\varepsilon_0}}) \cup (\mathcal{V}_\phi \times \Pi_{\bar{\mathcal{Q}}_{L,B,\varepsilon_0}})$ with $L = 1, B = \beta^2 + d\varepsilon_0$. According to Claim 3, $\mathcal{V}$ covers all possible value functions occurs when running Alg 2. We use $\mathcal{E}$ to denote the even that the following inequality holds for arbitrary $V \in \mathcal{V}$ and arbitrary $k \in [K] = [H], h \in [H]$:*

$$\Big\| \sum_{\tau=1}^{k-1} \sum_{n=1}^N \phi_h^{\tau n} \Big(V(s_{\tilde{h}+1}^{\tau n}) - \sum_{s' \in \mathcal{S}} P_h(s'|s_h^{\tau n}, a_h^{\tau n}) V(s') \Big) \Big\|_{(\Sigma_h)^{-1}} \leqslant c \cdot d\sqrt{\log(dN\beta H/\varepsilon_0\delta)}$$

As long as

$$\varepsilon_0 \leqslant 1/N \tag{25}$$

there exists some constant c, such that $P(\mathcal{E}) \geqslant 1 - \delta/2$.

Proof. We consider the value function class:

$$\mathcal{V} := \mathcal{V}_{L,B}^* \cup (\mathcal{V}_{L,B} \times \Pi_{\bar{\mathcal{Q}}_{L,B,\varepsilon_0}}) \cup (\mathcal{V}_\phi \times \Pi_{\bar{\mathcal{Q}}_{L,B,\varepsilon_0}})$$

and we have an ε_0-cover for it, which we denote as:

$$\mathcal{V}_{\varepsilon_0} := \bar{\mathcal{V}}_{L,B,\varepsilon_0}^* \cup (\bar{\mathcal{V}}_{L,B,\varepsilon_0} \times \Pi_{\bar{\mathcal{Q}}_{L,B,\varepsilon_0}}) \cup (\mathcal{V}_\phi \times \Pi_{\bar{\mathcal{Q}}_{L,B,\varepsilon_0}})$$

Besides, there exists $c' > 0$, s.t.

$$\log|\mathcal{V}_{\varepsilon_0}| \leqslant \log|\mathcal{V}_\phi \times \Pi_{\bar{\mathcal{Q}}_{L,B,\varepsilon_0}}| + \log|\bar{\mathcal{V}}_{L,B,\varepsilon_0}^*| + \log|\bar{\mathcal{V}}_{L,B,\varepsilon_0} \times \Pi_{\bar{\mathcal{Q}}_{L,B,\varepsilon_0}}|$$
$$\leqslant \log|\mathcal{V}_\phi \times \bar{\mathcal{Q}}_{L,B,\varepsilon_0}| + \log|\bar{\mathcal{V}}_{L,B,\varepsilon_0}^*| + \log|\bar{\mathcal{V}}_{L,B,\varepsilon_0} \times \bar{\mathcal{Q}}_{L,B,\varepsilon_0}|$$

$$\leqslant c' d^2 \log \frac{dHN\beta}{\varepsilon_0}$$

By plugging into Lemma E.3 and considering the union bound over $k \in [K]$ and $h \in [H]$ (note that $K = H$), we have with probability $1 - \delta/2$,

$$\Big\| \sum_{\tau=1}^{k-1} \sum_{n=1}^{N} \phi_h^{\tau n} \Big(V(s_{\tilde{h}+1}^{\tau n}) - \sum_{s' \in \mathcal{S}} P_h(s'|s_h^{\tau n}, a_h^{\tau n}) V(s') \Big) \Big\|^2_{(\Lambda_h^k)^{-1}} \leqslant c'' d^2 \log \frac{dHN\beta}{\varepsilon_0 \delta} + 8N^2 {\varepsilon_0}^2$$

When $\varepsilon_0 \leqslant \frac{1}{N}$, the first term will dominate, and there must exists c such that

$$\Big\| \sum_{\tau=1}^{k-1} \sum_{n=1}^{N} \phi_h^{\tau n} \Big(V(s_{\tilde{h}+1}^{\tau n}) - \sum_{s' \in \mathcal{S}} P_h(s'|s_h^{\tau n}, a_h^{\tau n}) V(s') \Big) \Big\|_{(\Lambda_h^k)^{-1}} \leqslant c \cdot d \sqrt{\log(dN\beta H/\varepsilon_0 \delta)}$$

□

E.5 Bias Analysis

Lemma E.5 (Overestimation in Alg 5)**.** *Suppose we choose*

$$\beta = c'_\beta d \sqrt{\log \frac{dHN}{\varepsilon_0 \delta}} \tag{26}$$

for some $c_\beta > 0$. During the running of Alg 2, on the condition 4.5 and on the event of $\mathcal{E}$ in Theorem E.4, which holds with probability $1 - \delta/2$, for arbitrary $h \in [H]$ and $\widetilde{h} \leqslant h - 1$, the parameter $w_{\widetilde{h}}$ and value function $V_{\widetilde{h}+1}$ occurs in Algorithm 5 should satisfy:

$$|\phi(s,a)^\top w_{\widetilde{h}} - \sum_{s' \in \mathcal{S}} P_{\widetilde{h}}(s'|s,a) V_{\widetilde{h}+1}(s')| \leqslant \beta \|\phi(s,a)\|_{\Sigma_{\widetilde{h}}^{-1}}$$

and

$$V_{\widetilde{h}}^*(s) \leqslant V_{\widetilde{h}}(s) \leqslant V_{\widetilde{h}}^*(s) + \mathbb{E}_\pi[\sum_{h'=\widetilde{h}}^{h} \beta \|\phi(s_{h'}, a_{h'})\|_{\Sigma_{h'}^{-1}}] \leqslant V_{\widetilde{h}}^*(s) + \beta\xi$$

Proof. The proof is mainly based on Theorem E.4, and the steps are similar to Lemma B.3 in (Jin et al., 2019) and Lemma 3.1 in (Wang et al., 2020b) and we omit here. □

Lemma E.6 (Bias Accumulation in Alg 5)**.** *On the induction condition 4.5 and on the events in Theorem E.4 which holds with probability $1 - \delta/2$, if*

$$\varepsilon_0 \leqslant \frac{\beta\xi}{2H} \tag{27}$$

in Algorithm 5, for arbitrary $\bar{R}$ generated, we have:

$$V_1^*(s_1; \bar{R}) - V_1^{\bar{\pi}}(s_1; \bar{R}) \leqslant 3\beta\xi$$

where recall that we use $V^\pi(s; \bar{R})$ to denote the value function with $\bar{R}$ as reward function.

Proof. We will use $\pi_h(\cdot) := \arg\max_a Q_h(\cdot, a)$ to denote the optimal policy w.r.t. the Q function without discretization, although we do not deploy it in practice. According to Lemma E.2, we should have $\max_{s,h} |Q_h(s, \pi(s)) - Q_h(s, \bar{\pi}(s))| \leqslant 2\varepsilon_0$ for arbitrary $h \in [h]$.

Recall that $\bar{\pi} = \bar{\pi}_1 \circ \bar{\pi}_2 ... \circ \bar{\pi}_h$

$$\begin{aligned}
& V_1^*(s_1) - V_1^{\bar{\pi}}(s_1) \leqslant V_1(s_1) - V_1^{\bar{\pi}}(s_1) && \text{(Lemma E.5; Overestimation)} \\
=& Q_1(s_1, \pi_1(s_1)) - Q_1^{\bar{\pi}}(s_1, \bar{\pi}_1(s_1)) \leqslant Q_1(s_1, \bar{\pi}_1(s_1)) - Q_1^{\bar{\pi}}(s_1, \bar{\pi}_1(s_1)) + 2\varepsilon_0 && \text{(Lemma E.2)} \\
=& \mathbb{E}_{s_1 \sim d_1, a_1 \sim \bar{\pi}_1}[\min\{\phi(s_1, a_1) w_1^\top + u_1(s_1, a_1), H\} - P_2 V_2(s_1, a_1) + P_2 V_2(s_1, a_1) - P_2 V_2^{\bar{\pi}}(s_1, a_1)] + 2\varepsilon_0 \\
\leqslant& \mathbb{E}_{s_1 \sim d_1, a_1 \sim \bar{\pi}_1, s_2 \sim P_2(\cdot|s_1, a_1)}[V_2(s_2) - V_2^{\bar{\pi}}(s_2)] + 2\beta \mathbb{E}_{s_1 \sim d_1, a_1 \sim \bar{\pi}_1}[\|\phi(s_1, a_1)\|_{\Sigma_1^{-1}}] + 2\varepsilon_0 \\
\leqslant& ...
\end{aligned}$$

$$\leqslant 2\beta\mathbb{E}_{s_1\sim d_1,a_1,s_2,\ldots,s_{h-1},a_{h-1}\sim\bar{\pi}}[\sum_{\widetilde{h}=1}^{h-1}\|\phi(s_{\widetilde{h}},a_{\widetilde{h}})\|_{\Sigma_{\widetilde{h}}^{-1}}]+2(h-1)\varepsilon_0+\mathbb{E}_{s_h,a_h\sim\bar{\pi}}[V_h(s_h)-V_h^{\bar{\pi}}(s_h)]$$

$$\leqslant 2\beta\xi+2h\varepsilon_0\leqslant 3\beta\xi \qquad \text{(Condition 4.5)}$$

□

Lemma E.7 (Bias of Linear Regression in Alg 6)**.** *During the running of Alg 2, on the event of $\mathcal{E}$ in Theorem E.4, which holds with probability $1-\delta/2$, for arbitrary $h\in[H]$, $\widetilde{h}\in[h-1]$, and arbitrary π, $\widehat{w}_{\widetilde{h}}^{\pi}$ and $\widehat{V}_{\widetilde{h}+1}^{\pi}$ occurs in Alg 6, we have:*

$$|\phi(s,a)^\top\widehat{w}_h^\pi-\sum_{s'\in\mathcal{S}}P_h(s'|s,a)\widehat{V}_{\widetilde{h}+1}^{\pi}(s')|\leqslant\beta\|\phi(s,a)\|_{\Sigma_h^{-1}}$$

where β is the same as Lemma E.5.

The proofs for the above Lemma is based on Theorem E.4 and Claim 3 and is similar to Lemma 3.1 in (Wang et al., 2020b), so we omit it here.

Lemma E.8 (Policy Evaluation Error in Alg 6)**.** *During the running of Algorithm 2, on the events of $\mathcal{E}$ in Theorem E.4, which holds with probability $1-\delta/2$, and on the induction condition in 4.5, for arbitrary $h\in[H]$ and $i,j\in[d]$, and arbitrary policy π and their evaluation results $\widehat{V}^\pi$ Algorithm 6, we have:*

$$|V^\pi(s_1;\widetilde{R}^{ij})-\widehat{V}_1^\pi(s_1)|\leqslant\beta\xi$$

where we use $\widetilde{R}^{ij}$ to denote the reward function used in Algorithm 6.

Proof. As a result of Lemma.E.7 , for arbitrary $\widetilde{R}^{ij}$, we have:

$$\begin{aligned}
|\widehat{V}_1^\pi(s_1)-V_1^\pi(s_1;\widetilde{R}^{ij})|=&|\widehat{Q}_1^\pi(s_1,a_1)-Q_1^\pi(s_1,a_1;\widetilde{R}^{ij})|\\
=&|\phi(s_1,a_1)^\top\widehat{w}_1^\pi-\sum_{s_2}P_h(s_2|s_1,a_1)V^\pi(s_2;\widetilde{R}^{ij})|\\
\leqslant&|\phi(s_1,a_1)^\top\widehat{w}_1^\pi-\sum_{s_2}P_h(s_2|s_1,a_1)\widehat{V}_2^\pi(s_2)|+\mathbb{E}_\pi|\widehat{V}_2^\pi(s_2)-Q_2^\pi(s_2,\pi(s_2);\widetilde{R}^{ij})|\\
\leqslant&\beta\|\phi(s_1,a_1)\|_{\Sigma_1^{-1}}+\mathbb{E}_{s_2}|V^\pi(s_2)-Q_2^\pi(s_2,\pi(s_2);\widetilde{R}^{ij})|\\
&\ldots\\
\leqslant&\beta\mathbb{E}_{s_1,a_1,\ldots,s_{h-1},a_{h-1}\sim\pi}[\sum_{t=1}^{h-1}\|\phi(s_t,a_t)\|_{\Sigma_t^{-1}}]\\
\leqslant&\beta\frac{h-1}{H}\xi\leqslant\beta\xi
\end{aligned}$$

□

E.6 Main Theorem and Proof

Now, we restate Theorem 4.4 in a formal version below:

Theorem E.9 (Formal Version of Theorem 4.4)**.** *For arbitrary $0<\varepsilon,\delta<1$, there exists absolute constants c_i,c_β and c_N, such that by choosing*

$$i_{\max}=c_i\frac{d}{\nu_{\min}^4}\log\frac{d}{\nu_{\min}},\quad \beta=c_\beta d\sqrt{\log\frac{dH}{\varepsilon\delta\nu_{\min}}},$$
$$N=c\Big(\frac{H^2d^3}{\varepsilon^2\nu_{\min}^2}+\frac{H^2d^7}{\nu_{\min}^{14}}\Big)\log^2\frac{dH}{\varepsilon\delta\nu_{\min}},\quad \varepsilon_0=\frac{1}{N}.$$

with probability $1-\delta$, after $K=H$ deployments, by running Alg 4 with the collected dataset $D=\{D_1,\ldots,D_H\}$ and arbitrary r satisfying the linear assumption in A (in Line of Algorithm 2), we will obtain a policy $\widehat{\pi}$ such that $V_1^\pi(s_1;r)\geqslant V_1^{\pi^*}(s_1;r)-\varepsilon$.

As additional guarantees, after h deployments, by running Alg 4 with the collected dataset $\{D_1, D_2..., D_h\}$ and reward function r, we will obtain a policy $\pi_{|h}$ which is ε-optimal in the MDP truncated at step h.

Proof. We will use $\Sigma_{h,i} := 2I + \sum_{j=1}^{i-1} \mathbb{E}_{\pi_j}[\phi(s_h, a_h)\phi(s_h, a_h)^\top]$ to denote the matrix which $\widetilde{\Sigma}_{h,i}$ approximates in Alg. 2, and use $R_{h,i} := \sqrt{\phi(\cdot,\cdot)^\top \Sigma_{h,i}^{-1} \phi(\cdot,\cdot)}$ to denote the reward function used in Alg 5 if the covariance matrix estimation is perfect (i.e. $\widetilde{\Sigma}_{h,i} = \Sigma_{h,i}$).

The proof consists of three steps. In step 1, we try to show that the inner loop of Alg 2 will terminate and Π_h will contain a set of exploratory policies. In step 2, we will analyze the samples generated by a mixture of policies in Π_h. In the last step, we determine the choice of hyper-parameters and fill the gaps of pre-assumed constraints and induction conditions.

Step 1: Exploration Ability for Policies in Π_h In the inner loop (line 5 - 12) in Algorithm 2, our goal is to find a set of policies Π_h, such that if the algorithm stops at iteration i, the following uncertainty measure is as small as possible

$$V^*_{h,i+1}(s_1; R_{h,i}) := \max_\pi \mathbb{E}_\pi[\|\phi(s_h, a_h)\|_{\Sigma_{h,i}^{-1}}] \tag{28}$$

To achieve this goal, we repeatedly use Alg 6 to estimate the covariance matrix of the policy and append it to $\widetilde{\Sigma}_{h,i}$ as an approximation of $\Sigma_{h,i}$, and use Alg 5 to find a near-optimal policy to maximizing the uncertainty-based reward function $\widetilde{R}$, by sampling trajectories with which we can reduce the uncertainty $Q^*_{h,i}$ in Eq.(28).

First, we take a look at the estimation error of the accumulative covariance matrix when running Algorithm 2. On the conditions in Lemma E.8, we can bound the elementwise estimation error of $\Sigma_{h,i}$:

$$|(\widetilde{\Sigma}_{h,i})_{jk} - (\Sigma_{h,i})_{jk}| \leqslant i \cdot \beta\xi, \quad \forall j, k \in [d]$$

As a result of Lemma E.11, we have:

$$\begin{aligned} |\widetilde{R}_{h,i}(s_h, a_h) - R_{h,i}(s_h, a_h)| =& |\sqrt{\phi^\top \widetilde{\Sigma}_{h,i}^{-1}\phi} - \sqrt{\phi^\top \Sigma_{h,i}^{-1}\phi}| \\ \leqslant& \sqrt{\frac{i \cdot d\beta\xi}{1 - i \cdot d\beta\xi}} \leqslant \sqrt{\frac{i_{\max} d\beta\xi}{1 - i_{\max} d\beta\xi}} \\ \leqslant& \frac{\nu^2_{\min}}{8} \end{aligned} \tag{29}$$

where the last but two step is because we at most repeat it $i_{\max}$ iterations at each layer h, and we introduce the following constraint for ξ during the derivation, to make sure the bias is small and all the iterms occurs in the derivation is well defined:

$$\xi \leqslant \frac{\nu^4_{\min}}{32 i_{\max} d\beta} \tag{30}$$

Next, we want to find a good choice of $i_{\max}$ to make sure $V_{h,i+1}$ will not always be large and the for-loop will break for some $i \leqslant i_{\max}$. We first provide an upper bound for $V_{h,i+1}$:

$$\begin{aligned} V_{h,i+1}(s_1) \leqslant& V^{\pi_{h,i+1}}(s_1; \widetilde{R}_{h,i}) + \beta\xi && \text{(Lemma E.5)} \\ \leqslant& V^{\bar{\pi}_{h,i+1}}(s_1; \widetilde{R}_{h,i}) + 4\beta\xi && (V^\pi - V^{\bar{\pi}} \leqslant V^* - V^{\bar{\pi}}; \text{Lemma E.6}) \\ \leqslant& V^{\bar{\pi}_{h,i+1}}(s_1; R_{h,i}) + 4\beta\xi + \frac{\nu^2_{\min}}{8} && \text{(bias of reward)} \\ \leqslant& V^{\bar{\pi}_{h,i+1}}(s_1; R_{h,i}) + \frac{\nu^2_{\min}}{4} && \text{(Constraints on } \xi \text{ in Eq.(30))} \end{aligned}$$

Next, we try to show that $V^{\bar{\pi}_{h,i+1}}(s_1; R_{h,i})$ can not always be large. According to Elliptical Potential Lemma in Lemma C.1, we have:

$$\sum_{i=1}^{i_{\max}} V^{\bar{\pi}_{h,i+1}}(s_1; R_{h,i}) = \sum_{i=1}^{i_{\max}} \mathbb{E}_{\bar{\pi}_{h,i+1}}[\|\phi(s_h, a_h)\|_{\Sigma_{h,i}^{-1}}]$$

$$
\begin{aligned}
&\leqslant \sum_{i=1}^{i_{\max}} \sqrt{\mathbb{E}_{\bar{\pi}_{h,i+1}}[\|\phi(s_h,a_h)\|^2_{\Sigma^{-1}_{h,i}}]} \\
&\leqslant \sqrt{i_{\max} \sum_{i=1}^{i_{\max}} \mathbb{E}_{\bar{\pi}_{h,i+1}}[\|\phi(s_h,a_h)\|^2_{\Sigma^{-1}_{h,i}}]} \\
&= \sqrt{i_{\max} \sum_{i=1}^{i_{\max}} Tr(\mathbb{E}_{\bar{\pi}_{h,i+1}}[\phi(s_h,a_h)\phi(s_h,a_h)^\top]\Sigma^{-1}_{h,i})} \\
&\leqslant \sqrt{2 i_{\max} d \log(1+i_{\max}/d)}
\end{aligned}
$$

where in the last step, we use the definition of $\Sigma_{h,i}$. Therefore,

$$
\min_i V^{\bar{\pi}_{h,i+1}}(s_1; R_{h,i}) \leqslant \frac{1}{i_{\max}} \sum_{i=1}^{i_{\max}} V^{\bar{\pi}_{h,i+1}}(s_1; R_{h,i}) \leqslant \sqrt{2\frac{d\log(1+i_{\max}/d)}{i_{\max}}}
$$

In order to guarantee $\min_i V^{\bar{\pi}_{h,i+1}}(s_1; R_{h,i}) \leqslant \nu^2_{\min}/8$, we require:

$$
\sqrt{2\frac{d\log(1+i_{\max}/d)}{i_{\max}}} \leqslant \nu^2_{\min}/8
$$

which can be satisfied by:

$$
i_{\max} = c_i \frac{d}{\nu^4_{\min}} \log \frac{d}{\nu_{\min}} \tag{31}
$$

for some absolute constant c_i.

Combining the above results, we can conclude that the inner loop in Alg 2 will break at some $i < i_{\max}$, such that $V_{h,i+1} \leqslant 3\nu^2_{\min}/8$, and guarantee that:

$$
\begin{aligned}
\max_\pi \mathbb{E}_\pi[\|\phi(s_h,a_h)\|_{\Sigma^{-1}_{h,i}}] :=& V^*_{h,i+1}(s_1; R_{h,i}) \\
\leqslant& V^*_{h,i+1}(s_1; \widetilde{R}_{h,i}) + \frac{\nu^2_{\min}}{8} && \text{(reward estimation error Eq.(29))} \\
\leqslant& V_{h,i+1}(s_1) + \frac{\nu^2_{\min}}{8} && \text{(Overestimation in Lemma E.5)} \\
\leqslant& V^{\bar{\pi}_{h,i+1}}(s_1; R_{h,i}) + \frac{\nu^2_{\min}}{4} + \frac{\nu^2_{\min}}{8} \\
\leqslant& \frac{\nu^2_{\min}}{2}
\end{aligned}
$$

Step 2: Policy Deployment and Concentration Error For uniform mixture policy $\pi_{h,mix} := Unif(\Pi_h)$, by applying Lemma E.13, Lemma E.14 and the results above, we must have:

$$
\begin{aligned}
&\max_\pi \mathbb{E}_\pi[\sqrt{\phi(s_h,a_h)^\top (I + N\mathbb{E}_{\pi_{h,mix}}[\phi\phi^\top])^{-1}\phi(s_h,a_h)}] \\
=& \max_\pi \mathbb{E}_\pi[\sqrt{\phi(s_h,a_h)^\top (I + \frac{N}{|\Pi_h|}|\Pi_h|\mathbb{E}_{\pi_{h,mix}}[\phi\phi^\top])^{-1}\phi(s_h,a_h)}] \\
\leqslant& \sqrt{\frac{2}{1+N/|\Pi_h|}} \max_\pi \mathbb{E}_\pi[\sqrt{\phi(s_h,a_h)^\top (I + |\Pi_h|\mathbb{E}_{\pi_{h,mix}}[\phi\phi^\top])^{-1}\phi(s_h,a_h)}] \\
\leqslant& \sqrt{\frac{1}{1+N/i_{\max}}}\nu_{\min} \leqslant \sqrt{\frac{i_{\max}}{N}}\nu_{\min}
\end{aligned}
$$

and this is the motivation of breaking criterion in Line 8 in Alg 2.

In the following, we will use $\Sigma_h^- := \Sigma_h - I = \sum_{n=1}^N \phi(s_{h,n}a_{h,n})\phi(s_{h,n},a_{h,n})^\top$ to denote the matrix of sampled feature without regularization terms, according to Lemma E.10, with probability $1-\delta/2$, we have:

$$\|\frac{1}{N}\sigma_{\max}(N\mathbb{E}_{\pi_{h,mix}}[\phi\phi^\top]-\Sigma_h^-)\| \leqslant \frac{4}{\sqrt{N}}\log\frac{8dH}{\delta}, \quad \forall h\in[H]$$

Follow the same steps in the proof of Lemma E.14, we know that

$$\sigma_{\min}(N\mathbb{E}_{\pi_{h,mix}}[\phi\phi^\top]) = \frac{N}{|\Pi_h|}\sigma_{\min}(|\Pi_h|\mathbb{E}_{\pi_{h,mix}}[\phi\phi^\top]) \geqslant \frac{N}{|\Pi_h|} \geqslant \frac{N}{i_{\max}}.$$

As a result,

$$\begin{aligned}\min_{x:\|x\|=1} x^\top\Sigma_h x = &\min_{x:\|x\|=1} x^\top(I+N\mathbb{E}_{\pi_{h,mix}}[\phi\phi^\top])x + x^\top(N\mathbb{E}_{\pi_{h,mix}}[\phi\phi^\top]-\Sigma_h^-)x\\ \geqslant&\sigma_{\min}(I+N\mathbb{E}_{\pi_{h,mix}}[\phi\phi^\top]) - \sigma_{\max}(N\mathbb{E}_{\pi_{h,mix}}[\phi\phi^\top]-\Sigma_h^-)\\ \geqslant&1+(\frac{N}{2i_{\max}} - 4\sqrt{N}\log\frac{8dH}{\delta})\end{aligned}$$

which implies that, as long as

$$N \geqslant 16 i_{\max}^2 \log^2\frac{8dH}{\delta} \tag{32}$$

we have

$$\sigma_{\max}(\Sigma_h^{-1}) \leqslant \frac{1}{1+N/2i_{\max}-\sqrt{N}\log\frac{8dH}{\delta}} \leqslant \frac{4i_{\max}}{N}$$

Therefore, for arbitrary π, we have:

$$\begin{aligned}&|\mathbb{E}_\pi[\sqrt{\phi(s_h,a_h)^\top(I+N\mathbb{E}_{\pi_{h,mix}}[\phi\phi^\top])^{-1}\phi(s_h,a_h)}] - \mathbb{E}_\pi[\sqrt{\phi(s_h,a_h)^\top(\Sigma_h)^{-1}\phi(s_h,a_h)}]|\\ \leqslant&\mathbb{E}_\pi[\sqrt{|\phi(s_h,a_h)^\top(I+N\mathbb{E}_{\pi_{h,mix}}[\phi\phi^\top])^{-1}\phi(s_h,a_h) - \phi(s_h,a_h)^\top(\Sigma_h)^{-1}\phi(s_h,a_h)|}]\\ \leqslant&\mathbb{E}_\pi[\sqrt{|\phi(s_h,a_h)^\top\Big((I+N\mathbb{E}_{\pi_{h,mix}}[\phi\phi^\top])^{-1}-(\Sigma_h)^{-1}\Big)\phi(s_h,a_h)|}]\\ =&\mathbb{E}_\pi[\sqrt{|\phi(s_h,a_h)^\top(I+N\mathbb{E}_{\pi_{h,mix}}[\phi\phi^\top])^{-1}\big(N\mathbb{E}_{\pi_{h,mix}}[\phi\phi^\top]-\Sigma_h^-\big)(\Sigma_h)^{-1}\phi(s_h,a_h)|}]\\ \leqslant&\sqrt{\sigma_{\max}((I+N\mathbb{E}_{\pi_{h,mix}}[\phi\phi^\top])^{-1})\frac{4i_{\max}}{N}\sigma_{\max}(N\mathbb{E}_{\pi_{h,mix}}[\phi\phi^\top]-\Sigma_h^-)}\\ \leqslant&\sqrt{\frac{1}{1+N/i_{\max}}\frac{16i_{\max}}{\sqrt{N}}\log\frac{8d}{\delta}} \qquad (i_{\max}\mathbb{E}_{\pi_{h,mix}}[\phi\phi^\top]\geq|\Pi_h|\mathbb{E}_{\pi_{h,mix}}[\phi\phi^\top]\geq 1)\\ \leqslant&\frac{4i_{\max}}{N^{3/4}}\sqrt{\log\frac{8dH}{\delta}}\end{aligned}$$

As a result,

$$\begin{aligned}&\max_\pi \mathbb{E}_\pi[\sqrt{\phi(s_h,a_h)^\top(\Sigma_h)^{-1}\phi(s_h,a_h)}]\\ \leqslant&\max_\pi \mathbb{E}_\pi[\sqrt{\phi(s_h,a_h)^\top(I+N\mathbb{E}_{\pi_{h,mix}}[\phi\phi^\top])^{-1}\phi(s_h,a_h)}] + \frac{4i_{\max}}{N^{3/4}}\sqrt{\log\frac{8dH}{\delta}}\\ \leqslant&\sqrt{\frac{i_{\max}}{N}}\nu_{\min} + \frac{4i_{\max}}{N^{3/4}}\sqrt{\log\frac{8dH}{\delta}}\end{aligned}$$

In order to make sure the induction conditions holds, we need

$$\sqrt{\frac{i_{\max}}{N}}\nu_{\min} + \frac{4i_{\max}}{N^{3/4}}\sqrt{\log\frac{8dH}{\delta}} \leqslant \xi/H$$

As long as we tighten the constraint in 32 to:

$$N \geqslant 256 \frac{i_{\max}^2}{\nu_{\min}^4} \log^2 \frac{8dH}{\delta} = \widetilde{O}(\frac{d^2}{\nu_{\min}^{12}}) \tag{33}$$

the induction conditions can be satisfied when

$$2\sqrt{\frac{i_{\max}}{N}} \nu_{\min} \leqslant \xi/2H$$

or equivalently,

$$N \geqslant \frac{16H^2 \nu_{\min}^2 i_{\max}}{\xi^2} = O(\frac{H^2 d}{\xi^2 \nu_{\min}^2})$$

Step 3: Determine Hyper-parameters

(1) Resolution ε_0 Recall that we still have a constraint for Z_h in (24)

$$I + \Sigma_R \geq \frac{{\varepsilon_0}^2}{4} I, \quad \frac{4}{{\varepsilon_0}^2} I \geq 2I + \Sigma_R$$

Since we already determined $i_{\max}$ in Eq.(31), also recall our constraints on ξ in (30) the above constraints for ε_0 can be satisfied as long as:

$$\varepsilon_0 \leqslant \sqrt{\frac{1}{i_{\max}}} \tag{34}$$

Combining all the constraints of ε_0, including (23), (25), (27) and (34), we conclude that:

$$\varepsilon_0 \leqslant \min\{\frac{1}{N}, \frac{\beta}{\sqrt{N+1}}, \frac{1}{\sqrt{i_{\max}}}, \frac{\beta\xi}{H}\} = \frac{1}{N}$$

(2) Induction error ξ Besides the constraint in (30), we need another one to make sure the quality of the final output policy. By applying Lemma D.3 for planning algorithm Alg. 2, if the induction condition (4.5) holds till $h \in [H]$, Alg. 4 will return us a policy $\widehat{\pi}$ such that:

$$V^* - V^\pi \leqslant 2\beta \max_\pi \mathbb{E}_\pi[\sum_{\widetilde{h}=1}^{H} \|\phi(s_{\widetilde{h}}, a_{\widetilde{h}})\|_{\Sigma_{\widetilde{h}}^{-1}}] \leqslant 2\beta\xi$$

To make sure $V^* - V^{\widehat{\pi}} \leqslant \varepsilon$, we require $\xi \leqslant \frac{\varepsilon}{2\beta}$, which implies that

$$\xi \leqslant \min\{\frac{\nu_{\min}^4}{32 i_{\max} d\beta}, \frac{\varepsilon}{2\beta}\}$$

Choice of N **and** β Since $\varepsilon_0 = \frac{1}{N}$, by plugging it into Eq.(26), we may choose β to be:

$$\beta = c''_\beta d \sqrt{\log \frac{dHN}{\delta}}$$

Now, we are ready to compute N. When $\xi = \frac{\varepsilon}{2\beta} \leqslant \frac{\nu_{\min}^4}{32 i_{\max} d\beta}$, we have:

$$N = O(\frac{H^2 d}{\xi^2 \nu_{\min}^2}) = O(\frac{H^2 d^3}{\varepsilon^2 \nu_{\min}^2}) \log \frac{dH}{\varepsilon\delta\nu_{\min}}$$

and otherwise, we have:

$$N = O(\frac{H^2 d}{\xi^2 \nu_{\min}^2}) = O(\frac{H^2 d^7}{\nu_{\min}^{14}}) \log \frac{dH}{\varepsilon\delta\nu_{\min}}$$

Combining the additional constraint to control the concentration error in Eq.(33), the total number of complexity would at the level:

$$N \geqslant c\Big(\frac{H^2 d^3}{\varepsilon^2 \nu_{\min}^2} + \frac{H^2 d^7}{\nu_{\min}^{14}}\Big) \log^2 \frac{dH}{\varepsilon\delta\nu_{\min}}$$

and therefore,

$$\beta = c_\beta d \sqrt{\log \frac{dH}{\delta\varepsilon\nu_{\min}}}$$

for some c_β.

Near-Optimal Guarantee Under the events in Theorem E.4, considering the failure rate of concentration inequality in Step 2, we can conclude that the induction condition holds for $h \in [H]$ with probability $1-\delta$. Combining our discussion about choice of ξ above, the probability that Alg 2 will return us an ε-optimal policy would be $1-\delta$.

The additional guarantee in Theorem E.9 can be directly obtained by considering the induction condition at layer $h \in [H]$. □

E.7 Technical Lemma

Lemma E.10 (Matrix Bernstein Theorem (Theorem 6.1.1 in (Tropp, 2015)))**.** *Consider a finite sequence $\{\mathbf{S}_k\}$ of independent, random matrices with common dimension $d_1 \times d_2$. Assume that*

$$\mathbb{E}\mathbf{S}_k = 0 \;\; and \;\; \|\mathbf{S}_k\| \leqslant L \;\; for\ each\ index\ k$$

Introduce the random matrix

$$\mathbf{Z} = \sum_k \mathbf{S}_k$$

Let $v(\mathbf{Z})$ be the matrix variance statistic of the sum:

$$\begin{aligned} v(\mathbf{Z}) &= \max\{\|\mathbb{E}(\mathbf{Z}\mathbf{Z}^*)\|, \|\mathbb{E}(\mathbf{Z}^*\mathbf{Z})\|\} \\ &= \max\{\|\sum_k \mathbb{E}(\mathbf{S}_k\mathbf{S}_k^*)\|, \|\sum_k \mathbb{E}(\mathbf{S}_k^*\mathbf{S}_k)\|\} \end{aligned}$$

Then,

$$\mathbb{E}\|\mathbf{Z}\| \leqslant \sqrt{2v(\mathbf{Z})\log(d_1+d_2)} + \frac{1}{3}L\log(d_1+d_2)$$

Furthermore, for all $t \geqslant 0$

$$\mathbb{P}\{\|\mathbf{Z}\| \geqslant t\} \leqslant (d_1+d_2)\exp\Big(\frac{-t^2/2}{v(\mathbf{Z})+Lt/3}\Big)$$

Lemma E.11 (Matrix Perturbation)**.** *Given a positive definite matrix $A \succ I$ and Δ satisfying $|\Delta_{ij}| \leqslant \varepsilon < 1/d$, define matrix $A_+ = A + \Delta$, then for arbitrary $\phi \in \mathbb{R}^d$ with $\|\phi\| \leqslant 1$, we have:*

$$A_+ \succ 0, \quad |\phi^\top(A_+ - A)\phi| \leqslant d\varepsilon, \quad |\phi^\top(A_+^{-1} - A^{-1})\phi| \leqslant \frac{d\varepsilon}{1-d\varepsilon}$$

which implies that

$$\|A_+\| \leqslant \|A\| + \|\Delta\| \leqslant \|A\| + \|\Delta\|_F \leqslant \|A\| + d\varepsilon, \quad \|A_+^{-1}\| \geqslant \|A^{-1}\| - \frac{d\varepsilon}{1-d\varepsilon}$$

Moreover,

$$|\|\phi\|_{A_+^{-1}} - \|\phi\|_{A^{-1}}| \leqslant \sqrt{|\|\phi\|^2_{A_+^{-1}} - \|\phi\|^2_{A^{-1}}|} \leqslant \sqrt{\frac{d\varepsilon}{1-d\varepsilon}}$$

Proof. First of all, easy to see that

$$\sigma_{\max}(\Delta) \leqslant \|\Delta\|_F \leqslant d\varepsilon$$

and therefore we have

$$\sigma_{\min}(A_+) = \min_{x:\|x\|=1} x^\top A x + x^\top \Delta x > 1 - d\varepsilon > 0.$$

where we use $\sigma_{\min}$ and $\sigma_{\max}$ to denote the smallest and the largest sigular value, respectively, and use $\|\cdot\|_F$ to refer to the Frobenius norm. Therefore,

$$|\phi^\top(A_+ - A)\phi| = |\phi^\top\Delta\phi| \leqslant d\varepsilon$$

and

$$|\phi^\top(A_+^{-1} - A^{-1})\phi| = |\phi^\top A_+^{-1}(A_+ - A)A^{-1}\phi| \leqslant \sigma_{\max}(A_+^{-1})\sigma_{\max}(A_+ - A)\sigma_{\max}(A^{-1}) \leqslant \frac{d\varepsilon}{1-d\varepsilon}$$

Moreover,

$$|\|\phi\|_{A_+^{-1}} - \|\phi\|_{A^{-1}}| \leqslant \sqrt{|\|\phi\|_{A_+^{-1}} - \|\phi\|_{A^{-1}}| \cdot |\|\phi\|_{A_+^{-1}} + \|\phi\|_{A^{-1}}|} = \sqrt{|\phi^\top(A_+^{-1} - A^{-1})\phi|} \leqslant \sqrt{\frac{d\varepsilon}{1-d\varepsilon}}$$

□

Next, we will try to prove that, with a proper choice of N, Algorithm 2 will explore layer h to satisfy the recursive induction condition.

Lemma E.12 (Random Matrix Estimation Error)**.** *Denote $\Lambda_h^\pi = \mathbb{E}_\pi[\phi\phi^\top]$. Based on the same induction condition 2, we have:*

$$|\|\phi\|_{(\Lambda^\pi)^{-1}} - \|\phi\|_{(\widehat{\Lambda}^\pi)^{-1}}| \leqslant \sqrt{\frac{d\varepsilon}{1-d\varepsilon}}, \quad \forall \|\phi\| \leqslant 1$$

Proof. Based on Lemma E.8, we have:

$$|\Lambda_{ij}^\pi - \widehat{\Lambda}_{ij}^\pi| \leqslant \frac{h-1}{H}\varepsilon \leqslant \varepsilon$$

and as a result of Lemma E.11, we finish the proof. □

Lemma E.13. *Given a matrix $A \succeq \lambda I$ with $\lambda > 0$, and ϕ satisfies $\|\phi\| \leqslant 1$, then we have:*

$$\phi^\top(cI + nA)^{-1}\phi \leqslant \frac{\lambda + c}{\lambda n + c}\phi^\top(cI + A)^{-1}\phi, \quad \forall n > 1, c > 0$$

Proof. Because $A \succeq \lambda I$, we have

$$\begin{aligned} cI + nA =& cI + \frac{c(n-1)}{\lambda + c}A + (n - \frac{c(n-1)}{\lambda + c})A \succeq \Big(1 + \frac{\lambda(n-1)}{\lambda + c}\Big)cI + (n - \frac{c(n-1)}{\lambda + c})A \\ =& \frac{\lambda n + c}{\lambda + c}\Big(cI + A\Big) \end{aligned}$$

Therefore,

$$\phi^\top(cI + nA)^{-1}\phi \leqslant \frac{\lambda + c}{\lambda n + c}\phi^\top(cI + A)\phi$$

□

Lemma E.14. *Given a matrix $A \succeq 0$, suppose $\max_\pi \mathbb{E}_\pi[\|\phi\|_{(cI+A)^{-1}}] \leqslant \widetilde{\varepsilon} \leqslant \nu_{\min}^2/(2\sqrt{c})$, where $c > 0$ is a constant, where $\nu_{\min}$ is defined in Definition 4.3, we have:*

$$\max_\pi \mathbb{E}_\pi[\|\phi\|_{(cI+nA)^{-1}}] \leqslant \sqrt{\frac{c+1}{\sqrt{c}(c+n)}\widetilde{\varepsilon}}$$

Proof. Because $\|\phi\|_{(cI+A)^{-1}} \leqslant \|\phi\|_{(cI)^{-1}} \leqslant 1/\sqrt{c}$, we must have:

$$\max_\pi Tr((cI + A)^{-1}\mathbb{E}_\pi[\phi\phi^\top]) = \max_\pi \mathbb{E}_\pi[\|\phi\|^2_{(cI+A)^{-1}}] \leqslant \frac{1}{\sqrt{c}}\max_\pi \mathbb{E}_\pi[\|\phi\|_{(cI+A)^{-1}}] \leqslant \frac{\widetilde{\varepsilon}}{\sqrt{c}}$$

Consider the SVD of $A = U^\top \Sigma U$ with $\Sigma = (\sigma_{ii})_{i=1,\dots,d}$ and $U = [u_1, u_2 \dots, u_d]$, then we have:

$$\forall \pi, \quad \frac{\widetilde{\varepsilon}}{\sqrt{c}} \geqslant Tr((cI + A)^{-1}\mathbb{E}_\pi[\phi\phi^\top]) = Tr((cI + \Sigma)^{-1}U^\top\mathbb{E}_\pi[\phi\phi^\top]U) = \sum_{i=1}^d \frac{\mathbb{E}_\pi[|\phi^\top u_i|^2]}{c + \sigma_{ii}}.$$

According to the Definition 4.3, we have:

$$\frac{\widetilde{\varepsilon}}{\sqrt{c}} \geqslant \frac{\max_\pi \mathbb{E}_\pi[|\phi^\top u_i|^2]}{c + \sigma_{ii}} \geqslant \frac{\nu_{\min}^2}{c + \sigma_{ii}}, \quad \forall i \in [d]$$

which implies that

$$\sigma_{ii} \geqslant \frac{\sqrt{c}\nu_{\min}^2}{\widetilde{\varepsilon}} - c \geqslant c, \quad \forall i \in [d]$$

By applying Lemma E.13 and assign $\lambda = c$, we have:

$$\max_\pi \mathbb{E}_\pi[\|\phi\|_{(cI+nA)^{-1}}] \leqslant \max_\pi \sqrt{\mathbb{E}_\pi[\|\phi\|^2_{(cI+nA)^{-1}}]} \leqslant \max_\pi \sqrt{\frac{2c}{c + cn}\mathbb{E}_\pi[\|\phi\|^2_{(cI+A)^{-1}}]} \leqslant \sqrt{\frac{2}{\sqrt{c}(1+n)}\widetilde{\varepsilon}}$$

□

E.8 More about our reachability coefficient

Recall the definition of reachability coefficient in (Zanette et al., 2020) is:

$$\min_{h\in[H]} \min_{\|\theta\|=1} \max_{\pi} |\mathbb{E}_\pi[\phi_h]^\top \theta|$$

Easy to see that, for arbitrary θ with $\|\theta\| = 1$, we have

$$\max_\pi \sqrt{\mathbb{E}_\pi[(\phi_h^\top \theta)^2]} \geqslant \max_\pi \mathbb{E}_\pi[|\phi_h^\top \theta|] \geqslant \max_\pi |\mathbb{E}_\pi[\phi_h^\top \theta]|$$

Therefore,

$$\nu_{\min} = \min_{h\in[H]} \nu_h = \min_{h\in[H]} \min_{\|\theta\|=1} \max_\pi \sqrt{\mathbb{E}_\pi[(\phi_h^\top \theta)^2]} \geqslant \min_{h\in[H]} \min_{\|\theta\|=1} \max_\pi |\mathbb{E}_\pi[\phi_h^\top \theta]|$$

Besides, according to the min-max theorem, ν_h is also lower bounded by $\sqrt{\max_\pi \sigma_{\min}(\mathbb{E}_\pi[\phi_h \phi_h^\top])}$, to see this,

$$\max_\pi \sigma_{\min}(\mathbb{E}_\pi[\phi_h \phi_h^\top]) = \max_\pi \min_{\|\theta\|=1} \theta^\top \mathbb{E}_\pi[\phi_h \phi_h^\top]\theta = \max_\pi \min_{\|\theta\|=1} \mathbb{E}_\pi[(\phi^\top \theta)^2] \leqslant \min_{\|\theta\|=1} \max_\pi \mathbb{E}_\pi[(\phi^\top \theta)^2]$$

In fact, the value of $\max_\pi \sigma_{\min}(\mathbb{E}_\pi[\phi_h \phi_h^\top])$ is also related to the "Well-Explored Dataset" assumption in many previous literature in offline setting (Jin et al., 2021b), where it is assumed that there exists a behavior policy such that the minimum singular value of the covariance matrix is lower bounded. Therefore, we can conclude that our reachability coefficient $\nu_{\min}$ is also lower bounded by, e.g. $\underline{c}$ in Corollary 4.6 in (Jin et al., 2021b).

F Extended Deployment-Efficient RL Setting

F.1 Sample-Efficient DE-RL

In applications such as recommendation systems, the value of N cannot exceed the number of users our system serves during a period of time. Therefore, as an interesting extension to our framework, we can revise the constraint (b) in Definition 2.1 and explicitly assign an upper bound for N. Concretely, we may consider the following alternatives: (b') The sample size $N \leqslant d^{c_1} H^{c_2} \varepsilon^{-c_3} \log^{c_4} \frac{dH}{\varepsilon\delta}$, where $c_1, c_2, c_3, c_4 > 0$ are some constant fixed according to the real situation. Under these revised constraints, the lower bound for K may be different.

In fact, given constraints in the form of $N \leqslant N_0$, our results in Section 4 already implies an upper bound for K, since we can emulate 1 deployment of our algorithm that uses a large $N > N_0$ by deploying the same policy for $\lceil N/N_0 \rceil$ times. However, this may result in sub-optimal deployment complexity since we are not adaptively updating our policy within those N/N_0 deployments. It would be an interesting open problem to identify the fine-grained trade-off between K and N_0 in such a setting.

F.2 Safe DE-RL

Monotonic Policy Improvement Constraint In many applications, improvement of service quality after policy update is highly desired, and can be incorporated in our formulation by adding an additional constraint into Def 2.1:

$$(c) \quad J(\pi_{i+1}) \geqslant J(\pi_i) - \varepsilon. \tag{35}$$

Because we require the deployed policy has substantial probability to visit unfamiliar states so that the agent can identify the near-optimal policy as shown in Def 2.1-(a), we relax the strict policy improvement with an small budget term $\varepsilon > 0$ for exploration.

Trade-off between Pessimism and Optimism The balance between satisfying two contradictory constraints: (a) and (c), implies that a proper algorithm should leverage both pessimism and optimism in face of uncertainty. In Algorithm 7, we propose a simple but effective mixture policy srategy, where we treat pessimistic and optimistic algorithm as black boxes and mix the learned policies with a coefficient α. One key property of the mixed policy is that:

Algorithm 7: Mixture Policy Strategy

for *k=1,2,...,K* **do**
  $D = \{D_1, D_2, ..., D_k\}$
  $\pi_{k,\text{pessim}} \leftarrow \text{PessimismBased\_OfflineAlgorithm}(D)$
  $\pi_{k,\text{optim}} \leftarrow \text{OptimismBased\_BatchExplorationStrategy}(D)$
  // Mix policy in trajectory level, i.e. w.p. $1-\alpha$, $\tau \sim \pi_{\text{pessim}}$; w.p. α, $\tau \sim \pi_{\text{optim}}$
  $\pi_k \leftarrow (1-\alpha)\pi_{k,\text{pessim}} + \alpha\pi_{k,\text{optim}}$
  $D_k = \{\tau_n \sim \pi_k, \forall n \in [N]\}$
end

Property F.1 (Policy improvement for mixture policy)**.**

$$J(\pi_k) - J(\pi_{k-1}) \geqslant J(\pi_{k,\text{pessim}}) - J(\pi_{k-1}) - O(\alpha) \tag{36}$$

As a result, as long as the offline algorithm (which we treat as a black box here) has some policy improvement guarantee, such as (Kumar et al., 2020; Liu et al., 2020; Laroche et al., 2019), then Eq.(36) implies a substantial policy improvement if α is chosen appropraitely. Besides, if we use Algorithms in Section 4.1 or 4.2, and collecting $\widetilde{N} = \Theta(N/\alpha)$ samples, the guarantees in Theorem 4.1 and Theorem 4.4 can be extended correspondingly. Therefore, Alg 7 will return us a near-optimal policy after K deployments while satisfying the safety constraint (c) in (35).