Multiple-policy High-confidence Policy Evaluation

Christoph Dann
Google Research

Mohammad Ghavamzadeh
Google Research

Teodor V. Marinov
Google Research

Abstract

In reinforcement learning applications, we often want to accurately estimate the return of several policies of interest. We study this problem, multiple-policy high-confidence policy evaluation, where the goal is to estimate the return of all given target policies up to a desired accuracy with as few samples as possible. The natural approaches to this problem, i.e., evaluating each policy separately or estimating a model of the MDP, do not take into account the similarities between target policies and scale with the number of policies to evaluate or the size of the MDP, respectively. We present an alternative approach based on reusing samples from on-policy Monte-Carlo estimators and show that it is more sample-efficient in favorable cases. Specifically, we provide guarantees in terms of a notion of overlap of the set of target policies and shed light on when such an approach is indeed beneficial compared to existing methods.

1 INTRODUCTION

Policy evaluation aims to estimate the return (expected sum of rewards) of a given policy and is one of the fundamental problems in reinforcement learning (Sutton and Barto, 1992). The on-policy version of this problem has a natural and very popular approach, simply collect several episodes with the policy of interest and estimate the return as the average sum of rewards achieved. This Monte-Carlo estimator can be readily augmented with confidence intervals through standard tools like Hoeffding's inequality. In contrast, the off-policy version, where we would like to estimate the return of a target policy from data that was collected with other policies, requires more sophisticated estimators and has been a very active field of study. Approaches based on importance-sampling, model-based estimation, doubly-robust estimators and marginalized importance sampling have been proposed and analyzed (Precup, 2000; Li et al., 2015; Gottesman et al., 2019; Yin and Wang, 2020; Uehara et al., 2021; Yin et al., 2021; Hao et al., 2021). The off-policy version of policy evaluation is particularly relevant in offline RL, for example, when we want to determine the return of a policy with high confidence from existing data before deploying it in a real system.

However, in many cases we have the ability to test a policy for a few iterations in the real system before we determine a full deployment, for example in the common practice of controlled experiments for recommendation systems (Gunawardana et al., 2012), where a recommendation policy is evaluated on a small random fraction of users. In those applications, we often have multiple policies that we would like to evaluate, e.g. arising from using different hyperparameters in offline RL training. Thus, our goal is to estimate the return for K different policies up to some desired accuracy ϵ with high confidence while collecting as few samples as possible. Our work provides a theoretical study of this problem setting, which we call multiple-policy high-confidence policy evaluation.

A simple solution is to treat multiple-policy policy evaluation as individual on-policy policy evaluation problems, collect a certain number of samples with each policy and use on-policy Monte-Carlo estimator on the samples collected with the respective policy. Albeit simple and robust, this approach scales linearly with the number of policies K which may be undesirable. One can show that this linear dependence on K is unavoidable in the worst-case, but there may be hope to improve it when the policies are "similar". For example, if all K policies take the same actions on many states they visit, then sharing data among their estimators may reduce the sampling requirement and yield sample-complexity that is sub-linear in K.

Our work provides a solution with sub-linear in K sample-complexity in favorable cases by identifying a suitable notion of dissimilarity between policies to evaluate (target policies), called *expected disagreement*, and providing guarantees that scale as a function of this dissimilarity instead of K. We obtain our main result under a sampling oracle assumption that generalizes that of a generative model.

Proceedings of the 26th International Conference on Artificial Intelligence and Statistics (AISTATS) 2023, Valencia, Spain. PMLR: Volume 206. Copyright 2023 by the author(s).

We then show how this oracle can be implemented in the absence of a generative model through an appropriate reprocessing phase while retaining favorable, albeit worse, guarantees. A key component of our work is a sampling strategy that couples the roll-out of the different target policies. This builds on an ideas similar to trajectory synthesis / stitching (Fonteneau et al., 2013; Sussex et al.; Wang et al., 2020b). However, while those works create synthetic trajectories from a given set of transition samples, we aim to collect transitions in the first place that can be synthesized into independent trajectories for each target policy.

We complement our main theoretical result with an in-depth discussion of our sampling strategy and its theoretical results. One may for example wonder whether this sampling strategy is actually necessary or whether it is sufficient for our results to simply collect data with a uniform mixture over target policies and use state-of-the-art off-policy estimators to produce the return of each target policy. We answer this in the negative by providing examples where a uniform mixture sampling approach requires $\tilde{\Theta}(K/\epsilon^2)$ samples but our sampling strategy only needs $\tilde{O}(1/\epsilon^2 + K)$, with a reduction by a factor of K in the dominant term. However, we also show that our sampling strategy is not fully instance-optimal by providing a carefully crafted example where an improvement by a factor $\sqrt{K}$ is possible. We hope that this discussion provides valuable insights on what quantities govern the sample-complexity of our problem setting and which problem instances are crucial to address by future work.

2 PRELIMINARIES AND NOTATIONS

Notations: We adopt the following notation. For any integer K, the set $\{1, \ldots, K\}$ is denoted by $[K]$, and $U([K])$ is the uniform distribution over this set. For any event E, $\mathbb{1}\{E\} = 1$ if event E occurs and $\mathbb{1}\{E\} = 0$, otherwise. For any set $\mathcal{X}$, we denote by $\Delta(\mathcal{X})$ the probability simplex over $\mathcal{X}$. We define the clipping operator as $\text{clip}[a|b] = a \cdot \mathbb{1}(a \geq b)$ for all $a, b \in \mathbb{R}$. Finally, we use $\tilde{O}$ for the big O notation up to poly-logarithmic factors.

We model the agent's interaction with the environment as a tabular finite-horizon MDP with state space $\mathcal{S}$, action space $\mathcal{A}$ and horizon H. The cardinalities of state- and action-space are $S = |\mathcal{S}|$ and $A = |\mathcal{A}|$. Without loss of generality, we assume that the MDP is layered, i.e., the state space $\mathcal{S}$ can be partitioned into H disjoint sets $\{\mathcal{S}_h\}_{h=1}^H$ and the only possible state transitions are between those in $\mathcal{S}_h$ to those in $\mathcal{S}_{h+1}$, for all $h \in [H]$. For convenience, we also assume that there is only one starting state s_1, i.e., $\mathcal{S}_1 = \{s_1\}$. We sometimes use s_h in our derivations to explicitly refer to the states in $\mathcal{S}_h$. Finally, we denote by $r : \mathcal{S} \times \mathcal{A} \to [0, 1]$ and $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$, the reward and transition kernels of the MDP, with $P(s'|s, a)$ being the probability of transitioning to state s' after taking action a in state s. We assume that the rewards are deterministic, however, our results can be easily extended to sub-Gaussian rewards. We also use the symbol $s_\perp$ for a terminal state, i.e., a state that the system remains there forever under any action and without receiving any reward. A policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ is defined as a deterministic or stochastic mapping from states to actions. If a policy π is deterministic, we use $\pi(s) = a$ to say that π takes action a in state s, and if it is stochastic, we use $\pi(a|s)$ as the probability with which π takes action a in state s. Policies are non-stationary in a finite-horizon MDP, i.e., $\pi = \{\pi_h\}_{h=1}^H$. However, since in a layered MDP, each policy π_h only operates on the states in $\mathcal{S}_h$, we can simply represent all policies $\pi_h, \ \forall h \in [H]$, by π.

For any policy π, we denote its value function by $v^\pi : \mathcal{S} \to [0, H]$. We overload the notation and use $v^\pi \equiv v^\pi(s_1)$, which we refer to as the return of policy π. We also use $\hat{v}^\pi$ to refer to any estimate of the return of policy π. Finally, we denote the state occupancy measure of π by $d^\pi : \mathcal{S} \to [0, 1]$, where $d^\pi(s') = \sum_{s \in \mathcal{S}_{h-1}} \sum_{a \in \mathcal{A}} d^\pi(s)\pi(a|s)P(s'|s, a), \ \forall s' \in \mathcal{S}_h$. We also overload this notation and use $d^\pi(s, a) = d^\pi(s)\pi(a|s)$ as the state-action occupancy of policy π.

3 PROBLEM SETTING & EXISTING APPROACHES

We study the problem of *multiple-policy high-confidence policy evaluation*, where we are given K target policies $\pi_1, \ldots, \pi_K \in \Pi$ and a failure probability $\delta \in (0, 1)$, and our goal is to produce confidence intervals $\{\mathfrak{I}_k\}_{k=1}^K$ for the return of these policies, $\{v^{\pi_k}\}_{k=1}^K$, that are required to guarantee

$$\mathbb{P}\left(\forall k \in [K] \colon v^{\pi_k} \in \mathfrak{I}_k\right) \geq 1 - \delta.$$

We distinguish two versions of this problem: **1)** In the *fixed-budget* version, we are allowed to interact with the MDP for a given number n of time-steps and the goal is to produce the smallest possible confidence intervals, and **2)** In the *fixed-accuracy* version, we are given a desired accuracy ϵ and the goal is to produce confidence intervals, each of size at most ϵ, after the least number of time-steps.

We will now review the existing approaches to this problem and discuss their strengths and limitations.

3.1 On-Policy Monte-Carlo Estimation

The simplest and perhaps most commonly used approach is MC estimation (Fonteneau et al., 2013). Here, we simply collect a certain number n_k of trajectories $\mathfrak{T}_{1,k}, \mathfrak{T}_{2,k}, \ldots, \mathfrak{T}_{n_k,k}$ with each target policy π_k and compute a return estimate as the average of the returns achieved in these trajectories, i.e.,

$$\hat{v}^{\pi_k} = \frac{1}{n_k} \sum_{i=1}^{n_k} \sum_{(s,a)\in \mathcal{T}_{i,k}} r(s,a).$$

With a simple application of Hoeffing's inequality,

$$|v^{\pi_k} - \hat{v}^{\pi_k}| \leq \sqrt{\frac{2H^2 \log(2/\delta_k)}{n_k}},$$

holds with probability at least $1-\delta_k$. Allocating the same number of trajectories and failure probability to each policy $n_k = \frac{n}{K}$ and $\delta_k = \frac{\delta}{K}$, shows that we can construct confidence intervals this way that shrink at a rate of

$$\sqrt{\frac{2H^2 K \log(2K/\delta)}{n}}. \tag{1}$$

Conversely, for the fixed-accuracy setting, if we want to achieve accuracy ϵ, then we will require

$$\frac{2H^2 K}{\epsilon^2} \log(2K/\delta) \tag{2}$$

samples. We can replace the dependency on H by a problem-dependent quantity using Bernstein's inequality, but this is orthogonal to the focus of this paper. Hence, we will generally keep H for ease of exposition. Our focus is on the dependency on the number of target policies K, as well as the size of the MDP (the number of states S and actions A). While the MC estimator has no dependency on the size of the MDP, it has to always suffer a K penalty. This is because no sample is shared among the estimators of the target policies.

3.2 Model-Based Estimation with Reward Bonuses

A common way to share samples is to first build a model of the environment and then use it to construct estimators by evaluating each target policy in the model. In fact, confidence bounds on value functions from empirical model estimates are the backbone of most regret-minimization algorithms in tabular MDPs. Constructing confidence-bounds using reward bonuses is the most common approach and has been shown to achieve minimax-optimal regret rate of order $\sqrt{SAT}\,\mathrm{poly}(H)$ (Azar et al., 2017; Zanette and Brunskill, 2019; Dann et al., 2019). Taking a reward bonus approach for evaluating the return of policy π_k gives a confidence interval of the form

$$\mathbb{E}_{\widehat{P},\pi_k}\left[\sum_{h=1}^{H} r(S_h, A_h)\right] \pm \mathbb{E}_{\widehat{P},\pi_k}\left[\sum_{h=1}^{H} b(S_h, A_h)\right],$$

where $\widehat{P}$ is the empirical transition model and $\mathbb{E}_{\widehat{P},\pi_k}[\cdot]$ is the expectation in this model when executing policy π_k. The bonuses have the form $b(s,a) = \sqrt{\frac{C(s,a)}{n(s,a)}} + o(1/\sqrt{n(s,a)})$, where $n(s,a)$ is the number of observations of state-action pair (s,a) used to construct the empirical model and $C(s,a)$ is some bound on the variance of the return from this state-action pair on. Unfortunately, even for a single policy and on-policy data collection, confidence intervals based on per-state reward bonuses only shrink at a rate of $\sqrt{S/n}$ in some MDPs. A simple example of such behavior can be found in Figure 1. In this example there exists a single starting state with only one action followed by S terminal states. The transition kernel is a uniform distribution over the S terminal states.

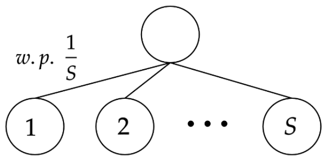

Figure 1: A simple MDP in which model-based estimators with reward bonuses have large confidence intervals.

3.3 Model-Based Estimation without Reward Bonuses

One may wonder whether the unavoidable S dependency is a limitation of the reward bonuses or the model-based estimator itself. Yin and Wang (2020) showed that the former is true by analyzing the root-mean-squared error of the model-based estimator $\mathbb{E}_{\widehat{P},\pi_k}\left[\sum_{h=1}^{H} r(S_h, A_h)\right]$ (they refer to as TMIS) and proving the following asymptotic convergence rate (Theorem 3.1 in their paper)

$$\sqrt{\frac{HK}{n}} \cdot \sqrt{\sum_{h=1}^{H} \mathbb{E}_{\pi_k}\left[\frac{d^{\pi_k}(s_h,a_h)}{\sum_{j=1}^{K} d^{\pi_j}(s_h,a_h)}\right]} + o\left(\frac{1}{\sqrt{n}}\right)$$

for policy π_k when the data is collected by a uniform mixture of all target policies. Since $d^{\pi_k}(s_h,a_h) \leq \sum_{j=1}^{K} d^{\pi_j}(s_h,a_h)$, this rate is indeed never worse than the MC rate in (1) and does not exhibit any explicit dependence on S. Further, when the occupancy of the target policies are similar, this rate can have more favorable dependency on K. However, using this result directly for confidence intervals is not straight-forward for several reasons: **1)** this is only a bound on the RMS, not a high probability bound, **2)** the bound includes the occupancy measures which are unknown, and **3)** the bound becomes valid only when the number of samples exceeds $\Theta\big(\max_{s,a}\left(\frac{1}{K}\sum_{j=1}^{K} d^{\pi_j}(s,a)\right)^{-1}\big)$. One can circumvent these issues by splitting the collected data in $\Theta(\log(1/\delta))$ batches, applying the estimators individually, and then constructing a confidence interval using the variation across batches (see Section 3.2 of Yin and Wang 2020), but such approach not always desirable in practice. Independent of the specific estimator, as we will see in Section 4.2, data collection with a uniform mixture of target policies can be highly suboptimal.

3.4 Other Estimators

Besides on-policy Monte Carlo and model-based estimators, there is a rich family of off-policy estimators for estimating the return of a target policy from a given dataset. A classic estimator is trajectory-wise or step-wise importance sampling but it is well-known that its variance is often exponential in the horizon (Liu et al., 2018, 2020). Doubly-robust estimators address this limitation (Jiang and Li, 2016; Hanna et al., 2017; Farajtabar et al., 2018) but, to the best of our knowledge, no finite-sample statistical bound is available for them that would allow us to construct provably accurate and tight confidence intervals. Recently, marginalized importance sampling with learned importance weights as the ratio of occupancy measures has been shown to be useful to overcome this curse of horizon. This includes several methods constructing confidence intervals (Jiang and Huang, 2020; Dai et al., 2020; Feng et al., 2021), with a focus on the function approximation case. However, to the best of our knowledge, these techniques only produce approximately correct intervals (Dai et al., 2020) or designed for the infinite-horizon case (Feng et al., 2021). Another line of recent work constructs confidence intervals for the off-policy optimization problem based on the α-Renyi divergence between occupancy measures of the target and collection policies (Papini et al., 2019; Metelli et al., 2020, 2021).

The above literature on off-policy policy evaluation provides a rich toolbox for policy evaluation from a fixed dataset but does not directly address the question of how to best sample the data in our setting, where we would like to evaluate a set of target policies. As we illustrate later, a simple uniform mixture of target policies is often suboptimal. In this work, we therefore explore a different approach inspired by the idea of trajectory synthesis that naturally yields a sampling strategy that performs better in natural problem instances.

4 MULTI-POLICY POLICY EVALUATION WITH A SAMPLING ORACLE

We study a simple procedure for the *fixed-accuracy* version of the multiple-policy high-confidence policy evaluation problem, whose pseudo-code is shown in Algorithm 1. The overall sample complexity of the approach is shown in Line 1. If the trajectories in Line 3 were sampled by executing each policy π_k for one episode, then Algorithm 1 would simply be on-policy Monte-Carlo (MC) estimation. As a result, the procedure inherits many of the benefits of on-policy MC estimation, including the simplicity and

```
Algorithm 1: High-Confidence Policy Evaluation
input: target policies $\{\pi_k\}_{k\in[K]}$,
       sampling oracle $\mathcal{O}$,
       failure probability $\delta$,
       desired accuracy $\epsilon$.
1 set number of trajectories to $n = \Theta\left(\frac{H^2 \log(K/\delta)}{\epsilon^2}\right)$
2 for $i = 1, 2, \ldots, n$ do
3     generate trajectories $\{\mathcal{T}_{i,k}\}_{k\in[K]}$ using
      Algorithm 2 with inputs $\{\pi_k\}_{k\in[K]}$ and $\mathcal{O}$
4 compute confidence intervals for all $k \in [K]$ as
  $\mathcal{I}_k = [\hat{v}^{\pi_k} - \epsilon - \beta_{\mathcal{O}}, \hat{v}^{\pi_k} + \epsilon + \beta_{\mathcal{O}}]$ with estimates
5     $\hat{v}^{\pi_k} = \frac{1}{n}\sum_{i=1}^{n}\sum_{(s,a)\in\mathcal{T}_{i,k}} r(s,a)$
6 return confidence intervals $\{\mathcal{I}_k\}_{k\in[K]}$
```

```
Algorithm 2: Coupled Trajectory Sampler
input: target policies $\{\pi_k\}_{k\in[K]}$, sampling oracle $\mathcal{O}$
1 initialize trajectories $\mathcal{T}_k = \varnothing, \ \forall k \in [K]$
2 initialize active sets
  $\mathcal{K}^1(s_1, a) = \{k \in [K] : \pi_k(s_1) = a\}$,
  $\mathcal{K}^1(s, a) = \varnothing, \quad \forall s \in \mathcal{S} \setminus \{s_1\}, \ \forall a \in \mathcal{A}$.
3 for $h \in [H]$ do
4     $\mathcal{K}^{h+1}(s, a) = \varnothing, \ \forall (s, a) \in \mathcal{S}_{h+1} \times \mathcal{A}$
5     for $(s, a) : \mathcal{K}^h(s, a) \neq \varnothing$ do
6         Query sampling oracle $s' = \mathcal{O}(s, a)$
7         for $k \in \mathcal{K}^h(s, a)$ do
8             Add $(s, a)$ to $\mathcal{T}_k$
9             if $s' \neq s_\perp$ then
10                Add $k$ to $\mathcal{K}^{h+1}(s', \pi_k(s'))$
11 return set of trajectories $\{\mathcal{T}_k\}_{k\in[K]}$
```

the opportunity to use empirical variance-based confidence bounds, which we omit here for ease of presentation.

The key component of the procedure is how the trajectories are generated which is shown in Algorithm 2. This procedure takes inspiration from Pegasus (Ng and Jordan, 2000) and Trajectory Synthesis Method in Wang et al. (2020a), and returns one trajectory for each target policy. However, instead of generating independent trajectories for each policy, as one would do in on-policy MC estimation, this procedure couples trajectories for different target policies by reusing parts of a trajectory for multiple policies (if this is possible). That is, if two policies play the same action at a visited state during a trajectory construction, then the trajectory follows the same successor state for both policies, instead of drawing a fresh sample for each policy. Assume for now that we have access to a generative model of the MDP and we can directly request successor state samples for a state-action pair. Then, if two policies would take the same initial action in the initial state, instead of requesting one sample for each, we only request one sample that can be used in the trajectories of both policies. Algorithm 2 follows this logic for all policies in all time-steps. The active set $\mathcal{K}^h(s,a) \subseteq [K]$ keeps track of which target policies visit state-action pair (s,a) at time h in their trajectories, and only requests one successor state sample per each non-empty active set. We will refer to $\mathcal{K}^h$ as the *arrangement* of policies at layer h.

We assume that we are given a sampling oracle that can generate the successor state samples for a given state-action pair. This may be a generative model if available, but as we will show in Section 5, can also be approximated when we only have trajectory access to the MDP. To be able to handle such approximate versions, we formally define the sampling-oracle as follows:

Definition 1. *A sampling oracle $\mathcal{O} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S} \cup \{s_\perp\} \times [c_{\max}])$ is a mapping from state-action pairs to a distribution over an extended state space $\mathcal{S} \cup \{s_\perp\}$ and a sample cost $m \in [c_{\max}]$, $c_{\max} \in \mathbb{N}$. Whenever the oracle $\mathcal{O}$ is called, an independent sample (s', m) from this distribution is returned that consists of the successor state s' and the cost m, where m is the total number of trajectories sampled from the MDP by the oracle to produce s'.*

The maximum cost $c_{\max}$ is an upper-bound on the total number of sampled trajectories that the oracle can use before returning $s' \in \mathcal{S} \cup \{s_\perp\}$. All the oracles we construct in Section 5 have $c_{\max} = O(K/\epsilon^2)$. As we will show, the dependence on $c_{\max}$ in our bounds is only logarithmic. Algorithm 2 uses a sampling oracle to collect trajectories for each target policy. Formally, a trajectory $\mathcal{T}$ sampled by an oracle $\mathcal{O}$ for a policy π is a sequence $\mathcal{T} = ((S_1, \pi(S_1)), \dots, (S_\tau, \emptyset))$, where $S_1 = s_1$ is always the starting state of the MDP and the trajectory ends whenever the oracle returns $s_\perp$ at the stopping time τ, i.e., $S_h = S_\tau$ iff $S_h = \mathcal{O}(s_{h-1}, \pi(s_{h-1}))$, $S_{h-1} \neq s_\perp$, and $S_h = s_\perp$. We further make the assumption that the sampling oracle returns $s_\perp$ almost surely whenever its input contains a state $s_H \in \mathcal{S}_H$ from the last layer of the MDP. A sampling oracle is characterized by its *bias*.

Definition 2. *The bias of the sampling oracle $\mathcal{O}$ with respect to a set of target policies Π is*

$$\beta_{\mathcal{O}} = \sup_{\pi \in \Pi} |\mathbb{E}[\hat{v}^\pi] - v^\pi| ,$$

where $\hat{v}^\pi = \sum_{(S_h, A_h) \in \mathcal{T}} r(S_h, A_h)$ and $\mathcal{T}$ is a trajectory sampled by oracle $\mathcal{O}$ for policy $\pi \in \Pi$.

The most trivial instantiation of the sampling oracle $\mathcal{O}$ is a generative model $\mathcal{O}(s,a) = P(\cdot|s,a)$ with $c_{\max} = 1$. The bias of this sampling oracle is $\beta_{\mathcal{O}} = 0$.

4.1 Theoretical Analysis

We first prove that Algorithm 1 is correct, i.e., the confidence intervals that it returns have the desired coverage.

Theorem 3. *If Algorithm 1 is called with parameters (ϵ, δ) and a sampling oracle $\mathcal{O}$ with bias $\beta_{\mathcal{O}}$, then with probability at least $1 - \delta/3$, the true values of all target policies $\{\pi_k\}_{k=1}^K$ are contained in the confidence intervals returned by the algorithm $\{\mathcal{I}_k\}_{k=1}^K$, i.e., $\mathbb{P}(\forall k \in [K] \colon v^{\pi_k} \in \mathcal{I}_k) \geq 1 - \delta/3$.*

Proof. We consider any target policy π_k, for a $k \in [K]$, and show that $\mathbb{P}(v^{\pi_k} \in \mathcal{I}_k) \geq 1 - (\delta/3K)$. The desired result follows after a union bound. Let $\hat{v}_i^{\pi_k} = \sum_{(s,a) \in \mathcal{T}_{i,k}} r(s,a)$ denote the empirical value of policy π_k constructed after sampling the i-th trajectory from the oracle. Consider the martingale difference sequence $\{\hat{v}_i^{\pi_k} - \mathbb{E}[\hat{v}_i^{\pi_k}]\}_{i=1}^n$, where the expectation is w.r.t. all the randomness up to the i-th call to the oracle. Using the Azuma-Hoeffding's inequality, with probability $1 - \delta$, we have

$$\left| \frac{1}{n} \sum_{i=1}^n \hat{v}_i^{\pi_k} - \mathbb{E}[\hat{v}_i^{\pi_k}] \right| \leq 2H \sqrt{\frac{\log(2/\delta)}{n}} .$$

Using the fact that $\mathcal{O}$ has bias $\beta_{\mathcal{O}}$, we have that

$$\left| v^{\pi_k} - \frac{1}{n} \sum_{i=1}^n \hat{v}_i^{\pi_k} \right| \leq \left| v^{\pi_k} - \frac{1}{n} \sum_{i=1}^n \mathbb{E}[\hat{v}_i^{\pi_k}] \right| + \left| \frac{1}{n} \sum_{i=1}^n \hat{v}_i^{\pi_k} - \mathbb{E}[\hat{v}_i^{\pi_k}] \right| \leq \beta_{\mathcal{O}} + 2H \sqrt{\frac{\log(6K/\delta)}{n}},$$

holds with probability at least $1 - \frac{\delta}{3K}$. The result follows by the definitions of n and $\mathcal{I}_k$ in Algorithm 1. □

After establishing the correctness of Algorithm 1, we now analyze its sample-complexity by bounding the total number of calls to the sampling oracle $\mathcal{O}$. We express our bound in terms of the *expected disagreement* of the policies, which we define next.

Definition 4 (Disagreement)**.** *The disagreement of the policy arrangement $\mathcal{K}^h$ is the number of state-action pairs visited by the target policies at layer h, i.e.,*

$$\operatorname{dis}(\mathcal{K}^h) = |\{(s,a) \in \mathcal{S}_h \times \mathcal{A} \colon \mathcal{K}^h(s,a) \neq \varnothing\}| .$$

The expected disagreement of $\mathcal{K}^h$ is defined as

$$\mathcal{D}(\mathcal{K}^h) = \mathbb{E}\left[\sum_{h'=h}^H \operatorname{dis}(\mathcal{K}^{h'}) \mid \mathcal{K}^h \right] .$$

Note that for each element in $\{(s,a) \in \mathcal{S}_h \times \mathcal{A} \colon \mathcal{K}^h(s,a) \neq \varnothing\}$, Algorithm 2 samples once from the sampling oracle $\mathcal{O}$. Hence, the expected number of samples used by Algorithm 1 is $n\mathcal{D}(\mathcal{K}^1)$. We can also show in the following theorem that the total number of samples used by Algorithm 1 concentrates tightly around the expected number of samples.

Theorem 5. *If Algorithm 1 is called with parameters (ϵ, δ), then with probability at least $1-\delta/3$, the total number of calls to the sampling oracle is bounded by*

$$O\left(\frac{\mathcal{D}(\mathcal{K}^1)\log(K/\delta)}{\epsilon^2} + KH\log(1/\delta)\right).$$

Proof of Theorem 5. The total number of oracle calls can be written as $\sum_{i=1}^n \sum_{h=1}^H \operatorname{dis}(\mathcal{K}_i^h)$. Writing $X_i = \sum_{h=1}^H \operatorname{dis}(\mathcal{K}^h) \in [0, KH]$ and noting that $\{X_i\}_{i\in[n]}$ is a set of i.i.d. random variables, we can apply Bernstein's inequality and bound the total number of oracle calls as

$$\sum_{i=1}^n \sum_{h=1}^H \operatorname{dis}(\mathcal{K}_i^h) \le \sum_{i=1}^n \mathbb{E}\left[\sum_{h=1}^H \operatorname{dis}(\mathcal{K}_i^h)\right] + c_1\sqrt{\sum_{h=1}^n \operatorname{Var}\left(\sum_{h=1}^H \operatorname{dis}(\mathcal{K}_i^h)\right)\log(3/\delta)} + c_2 HK\log(3/\delta)$$

with probability at least $1-\delta/3$ for appropriate absolute constants $c_1, c_2 > 0$. We further bound the RHS as

$$\begin{aligned}
&\le \sum_{i=1}^n \mathcal{D}(\mathcal{K}_i^1) + c_1\sqrt{KH\sum_{h=1}^n \mathbb{E}\left(\sum_{h=1}^H \operatorname{dis}(\mathcal{K}^h)\right)\log(3/\delta)} \\
&\quad + c_2 HK\log(3/\delta) \\
&= \sum_{i=1}^n \mathcal{D}(\mathcal{K}_i^1) + c_1\sqrt{KH\sum_{i=1}^n \mathcal{D}(\mathcal{K}_i^1)\log(3/\delta)} \\
&\quad + c_2 HK\log(3/\delta) \\
&\le \frac{3}{2}\sum_{i=1}^n \mathcal{D}(\mathcal{K}_i^1) + \left(\frac{c_1}{2} + c_2\right) HK\log(3/\delta)
\end{aligned}$$

where the first inequality uses $\operatorname{Var}(X) \le \mathbb{E}[X]\cdot \operatorname{esssup} X$ which holds for all non-negative random variables and the final inequality applies the arithmetic-mean-geometric-mean inequality. Plugging in the value of n and noting that $\mathcal{K}_i^1 = \mathcal{K}^1$ finishes the proof. □

Up to a small number[1] of samples $\tilde{O}(KH)$ that is independent of ϵ, the sample-complexity of Algorithm 1 is governed by that of the sampling oracle and the expected disagreement $\mathcal{D}(\mathcal{K}^1)$. The expected disagreement, which we defer its in-depth discussion to the next section, is a quantity that depends on the problem and target-policies, is favorably small in many problems, and more importantly is never larger than KH.

[1]Note that this number can be reduced to $c \ll HK$ if the disagreement of each $\mathcal{K}_i^h$, $\forall i \in [n]$, $\forall h \in [H]$ is uniformly bounded by c with probability $1-\delta$. We expect that if the target policies are similar, then such a uniform bound would be a reasonable assumption.

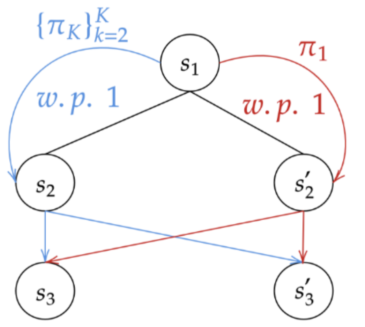

Figure 2: The class of MDPs used in Lemma 7.

When we have access to a generative model, Theorem 3 and Theorem 5 imply the following sample-efficiency guarantee for Algorithm 1:

Corollary 6. *If Algorithm 1 is called with parameters (ϵ, δ) and a generative model as its sampling oracle, then after requesting at most $O\left(\frac{\mathcal{D}(\mathcal{K}^1)\log(K/\delta)}{\epsilon^2} + KH\log(1/\delta)\right)$ samples from the generative model, with probability at least $1-\delta$, all confidence bounds it returns have width at most 2ϵ and capture the true value of all target policies.*

4.2 Discussion on the Expected Disagreement

While the notion of disagreement is somewhat natural and the coupled trajectory sampler (Algorithm 2) takes advantage of a natural notion of overlap between trajectories, it is unclear if Algorithm 1 is optimal even when it uses a generative model as its sampling oracle. Using two examples depicted in Figures 2 and 3, we demonstrate the benefits and limitations of Algorithm 1. The formal construction for the example in Figure 2 can be found in the proof of Lemma 7 and the construction for the example in Figure 3 can be found in the proof of Lemma 8 in Appendix A. We begin with a very simple MDP in which any round-robin-based sampling strategy will perform poorly compared to Algorithm 1:

Lemma 7. *Let $K \ge 2$ and $\epsilon = O(1/K)$ be arbitrary. Then, there is a class of MDPs and K target policies so that the expected disagreement $\mathcal{D}(\mathcal{K}^1) \le 2$ is constant and Algorithm 1 only requires*

$$O\left(\frac{\log(K/\delta)}{\epsilon^2} + KH\log(1/\delta)\right)$$

samples. However, any algorithm that collects a fixed-size dataset with the uniform mixture over all target policies has to collect at least $\Omega(K/\epsilon^2)$ samples.

Figure 2 shows the topology of the class of MDPs used in Lemma 7. The idea behind it is very simple. In the initial state, policies $\pi_2, \ldots \pi_K$ all take the same action that transitions deterministically to s_2 while π_1 takes a different action that transitions to s_2'. In both successor states,

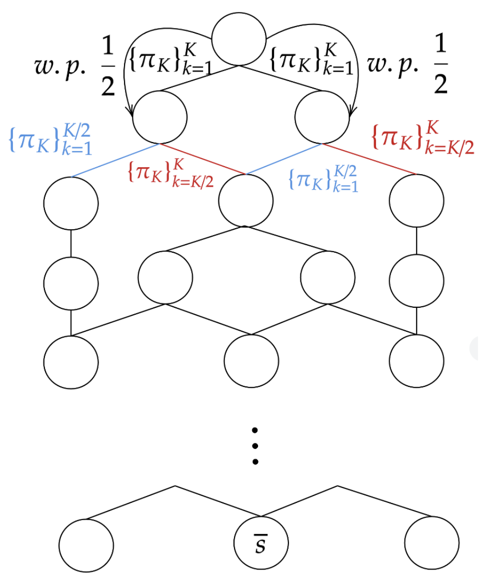

Figure 3: The MDP instance used for the lower-bound in Lemma 8.

each action transitions with almost equal probability to s_3 or s_3' and the only non-zero reward is in s_3 which is 1. Sampling with a uniform mixture over policies $\pi_1, \ldots, \pi_K$ will generate roughly K times more samples for s_2 than for s_2'. Since we need order $1/\epsilon^2$ samples for each s_2 and s_2', this will result in a total sample size of order K/ϵ^2. In contrast, the expected disagreement in this MDP is $O(1)$ and Algorithm 1 will generate roughly equal amount of samples for s_2 and s_2'. This gives a total expected sample size of order $1/\epsilon^2$. This example illustrates that even in very simple cases, the sampling strategy of Algorithm 1 can be much more sample-efficient than the simple uniform mixture over target policies.

There are also examples where Algorithm 1 performs worse than collecting data with a uniform mixture over target policies, but they require a more intricate setup than Figure 2. The next lemma shows such an example.

Lemma 8. *There exists an MDP instance with at most $O(\log(K))$ states and $O(K)$ actions for which*

$$\mathbb{E}\left[\sum_{h=1}^{H} \operatorname{dis}(\mathcal{K}^h)\right] \geq \Omega(\sqrt{K}) ,$$

$$\sum_{h=1}^{H}\sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}} \max_{k\in[K]} d_h^{\pi_k}(s,a) \leq O(\log(K)) .$$

Figure 3 shows the topology of the MDP instance used by Lemma 8. The first statement in Lemma 8 implies that Algorithm 1 will need to call the generative model at least $\Omega(\sqrt{K})$ times with constant probability, and we will show that the second implies that a model-based estimator with data collected by a uniform mixture over $\pi_1, \ldots, \pi_K$ will only require samples that are independent of K. To that end, we compute the accuracy of the round-robin model-based estimator for any policy π. Let $\hat{d}^\pi(s,a)$ denote the empirical occupancy measure computed after n rounds of round-robin sampling. We now bound $|\hat{d}^\pi(\bar{s},a) - d^\pi(\bar{s},a)|$ for the state $\bar{s}$ in Figure 3, where $a = d^\pi(\bar{s})$. Applying the Bernstein's inequality implies that

$$\mathbb{P}\left(|\hat{d}^\pi(\bar{s},a) - d^\pi(\bar{s},a)| \gtrsim \sqrt{\frac{d^\pi(\bar{s},a)}{n(\bar{s},a)}\log(1/\delta)}\right) \leq \delta.$$

The construction of the MDP instance implies that $d^\pi(\bar{s},a) = \Theta(1/\sqrt{K})$. Thus, another application of Hoeffding's inequality implies that $n(\bar{s},a) = \Omega(\sqrt{K}n)$ with probability $1-\delta$. Combining with the above Bernstein bound, we have that $|\hat{d}^\pi(\bar{s},a) - d^\pi(\bar{s},a)| < O(\sqrt{\frac{\log(1/\delta)}{n}})$ with probability $1-\delta$. Similarly, we can show that for any state-action pair (s,a), it holds that $|\hat{d}^\pi(s,a) - d^\pi(s,a)| < O(\sqrt{\frac{\log(1/\delta)}{n}})$ and so a uniform bound over all (s,a)'s shows that $|\hat{v}^\pi - v^\pi| \leq O(\sqrt{\frac{\log(SA/\delta)}{n}})$ with probability $1-\delta$. This implies that the sample complexity of a simple model-based estimator is $\sqrt{K}$ times less than that of Algorithm 1.

5 SAMPLING ORACLE WITHOUT A GENERATIVE MODEL

In this section, we discuss how we can implement a sampling oracle without access to a generative model. This will require a preprocessing phase that first identifies a relevant subset of states $\mathcal{G}$ and then for each state $s \in \mathcal{G}$ determines a policy to reach s. Equipped with the mapping of relevant states to policies $g\colon \mathcal{G} \to \Pi$, we can implement the sampling oracle $\mathcal{O}_g$ shown in Algorithm 3. When the oracle is invoked for a state-action pair (s,a) with $s \in \mathcal{G}$, we execute the policy $g(s)$ associated with s until s is reached, then we take action a and obtain a successor state sample. If either $s \notin \mathcal{G}$ or it takes too many attempts to reach s, then the oracle returns $s_\perp$ instead.

The preprocessing phase is shown in Algorithm 4. We first sample a hold-out set of trajectories $\mathcal{H}_k$ for each policy π_k. We then proceed in phases indexed by i. In each phase we again sample a dataset of trajectories for each policy and determine the empirical occupancy $\hat{d}_k$ for each policy π_k, but zero out states that did not occur often enough in the dataset. For all non-zero entries, we can show that $\hat{d}_k(s)$ is a constant factor approximation of the true occupancy measure d^{π_k}. We then determine the good set $\mathcal{G} \subseteq \mathcal{S}$ as all states for which there is at least one non-zero occupancy measure estimate. We also associate in $g(s)$ each state in $s \in \mathcal{G}$ with the policy that has the largest occupancy measure estimate for s. The mapping g now identifies for each state in $\mathcal{G}$ the policy that reaches each state with the high-

Algorithm 3: Sampling oracle $\mathcal{O}_g$

```
parameter: policy mapping $g\colon \mathcal{G} \to \Pi$
           maximum sample cost $c_{\max}$
input    : state $s$ and action $a$
1 if $s \notin \mathcal{G}$ then
2     return $s_\perp$
3 run policy $\pi = g(s)$ until either $s$ is reached or the
  number of episodes exceeds $c_{\max}$
4 if $s$ is reached then
5     execute action $a$
6     return next state $s'$
7 else
8     return $s_\perp$
```

Algorithm 4: Preprocessing phase

```
input: Policy $\pi$, confidence parameter $\delta$, accuracy $\epsilon$
1 initialize $c \leftarrow O(\log^2(SK/(\epsilon\delta)))$
2 for $k = 1, \dots K$ do
3     $\mathcal{H}_k \leftarrow$ Sample $\frac{10}{\epsilon}\log(c/\delta)$ trajectories with $\pi_k$
4 for $i = 0, 1, 2, \dots,$ do
5     for $k = 1, \dots, K$ do
6         Sample $2^i$ trajectories with $\pi$ and let
            $w_k\colon \mathcal{S} \to \mathbb{N}$ be the number of visits to each
            state
7         Set $\hat{d}_k(s) \leftarrow \frac{w_k(s)}{2^i}\mathbf{1}\left\{w_k(s) \geq 2\log\frac{c}{\delta}\right\} \forall s$
8     Set $\mathcal{G} \leftarrow \{s \in \mathcal{S}\colon \max_k \hat{d}_k(s) > 0\}$
9     for $s \in \mathcal{G}$ do
10        Set $g(s) \to \pi_{k'}$ where
11            $k' = \operatorname{argmax}_{k\in[K]} \hat{d}_k(s)$
12    Set $\tau_k$ as the number of trajectories in $\mathcal{H}_k$ that
        contain a state outside of $\mathcal{G}$
13    if $\tau_k \leq \frac{10}{3}\log\frac{c}{\delta}$ for all $k \in [K]$ then
14        return $g\colon \mathcal{G} \to \Pi$
```

est probability, up to a constant factor. All that is left is to ensure that $\mathcal{G}$ is sufficiently large. To that end, we test the escape probability of each policy π_k from the set $\mathcal{G}$, that is, the likelihood that a trajectory sampled with π_k contains a state outside of $\mathcal{G}$. We use the hold-out set $\mathcal{H}_k$ and if the number of escaped trajectories in $\mathcal{H}_k$ is sufficiently small, then the algorithm terminates and returns g. Otherwise, it moves on the to the next phase where it collects more trajectories to determine $\mathcal{G}$.

The following lemma formalizes the properties of this procedure:

Lemma 9. *With probability $1 - O(\delta)$, Algorithm 4 terminates after collecting*

$$O\left(\frac{SK}{\epsilon}\log\frac{\log(SK/\epsilon)}{\delta}\right)$$

episodes and the function $g\colon \mathcal{G} \to \Pi$ satisfies the following properties:

1. *The escape probability of any policy π_k for $k \in [K]$ from the set $\mathcal{G}$ is at most ϵ, that is, an episode sampled with π_k contains with probability at least $1-\epsilon$ only states in $\mathcal{G}$.*

2. *For each $s \in \mathcal{G}$, it holds*

$$d^{g(s)}(s) \geq \frac{1}{36}\max_{k\in[K]} d^{\pi_k}(s).$$

Note that the S factor in the number of collected episodes can be replaced by an instance-dependent factor that is much smaller than S in favorable cases. This preprocessing phase is sufficient to guarantee that the oracle in Algorithm 3 has a bias that is at most H times larger than the accuracy parameter ϵ:

Lemma 10. *Assume that Algorithm 3 receives a $g\colon \mathcal{G} \to \Pi$ such that the escape probability for all $\pi_1, \dots, \pi_k$ from $\mathcal{G}$ is at most ϵ. Then the bias of the sampling oracle is at most $H\epsilon$.*

Hence, by first running Algorithm 4 with accuracy parameter $\epsilon/2H$ and then using Algorithm 3 as a sampling oracle in Algorithm 1, we can achieve the following guarantee:

Theorem 11. *If Algorithm 1 is called with parameters $(\epsilon/2, \delta)$ and a sampling oracle $\mathcal{O}_g$ from Algorithm 3 with preprocessing in Algorithm 4, then with probability at least $1 - O(\delta)$, the return of all target policies $\{\pi_k\}_{k=1}^K$ is contained in the confidence intervals produced by the algorithm. Furthermore, these intervals have each size at most ϵ and the total number of trajectories collected is*

$$\tilde{O}\left(\frac{H^2}{\epsilon^2}\mathbb{E}\left[\sum_{(s,a)\in\mathcal{K}^{1:H}}\frac{1}{d^{\max}(s)}\right] + \frac{SH^2K}{\epsilon}\right)$$

where $d^{\max} = \max_{k\in[K]} d^{\pi_k}$ and we use $(s,a) \in \mathcal{K}^{1:H}$ to mean the sum over all state-action pairs with non-empty entry in $\mathcal{K}^{1:H}$ and $\mathcal{K}^{1:H}$ is the random arrangement produced by the coupled trajectory sampler

To provide some intuition about the bound in Theorem 11, we now compute the bound for the MDPs in Figure 2 and Figure 3. In Figure 3, by construction every policy has the same probability of visiting $\bar{s}$, i.e., $d^{\max}(\bar{s}) = \Theta(1/\sqrt{K})$. The disagreement is constant everywhere outside $\bar{s}$ and every arrangement has the same probability. So, the upper-bound in Theorem 11 evaluates to $\Theta(\mathcal{L}K/\epsilon^2)$ providing no improvement over just playing the round-robin strategy together with a MC estimator. In Figure 2, $d^{\max}(s) = \Omega(1)$ for all states and there exists only a single possible arrangement with disagreement $O(1)$. Thus, Theorem 11 implies a total sample complexity of $\tilde{O}\left(\frac{1}{\epsilon^2} + \frac{K}{\epsilon}\right)$

that matches the one by Algorithm 1 with the generative model oracle and improves w.r.t. K over both the MC and round-robin model-based estimators.

During the preprocessing phase in Algorithm 4, we essentially look for each state $s \in \mathcal{G}$ for the policy among $\pi_1, \dots, \pi_K$ which visits that states with highest probability. Using the ideas of reward-free exploration, we can search among a larger set of policies for the best one to visit each state. This trades of better sampling cost in the main phase against a highest sampling cost during preprocessing. We will then replace $d^{\max}(s)$ in the bound by

$$\mathfrak{d}(s) = \max_{\pi \in \Pi_{K,\mathcal{G}}} d^\pi(s),$$

which is the maximum occupancy measure over all policies $\Pi_{K,\mathcal{G}}$ restricted to the set of states $\mathcal{G}$ and actions taken by at least one target policy. It is sufficient here to consider the MDP that is restricted to the set of states $\mathcal{G}$ (produced by Algorithm 4) since this will contribute at most a bias of $O(\epsilon)$ in the final return estimate, as we have shown above.

Theorem 12. *There exists a sampling oracle whose total number of sampled trajectories when is used with Algorithm 1 is at most*

$$\tilde{O}\left(\frac{H^2}{\epsilon^2}\mathbb{E}\left[\sum_{(s,a)\in\mathcal{K}^{1:H}} \frac{1}{\mathfrak{d}(s)}\right] + \frac{KH^4S^4}{\epsilon}\right),$$

with probability $1-\delta$. Furthermore, the bias of this sampling oracle is at most $O(\epsilon)$.

The definition of $\mathfrak{d}(s)$ implies $\mathfrak{d}(s) \geq d^{\max}(s)$. We now discuss a simple MDP in which the bound in Theorem 11 evaluates to $\tilde{O}(K/\epsilon^2)$, while the improved bound in Theorem 12 evaluates to $\tilde{O}(1/\epsilon^2)$, matching the generative model oracle.

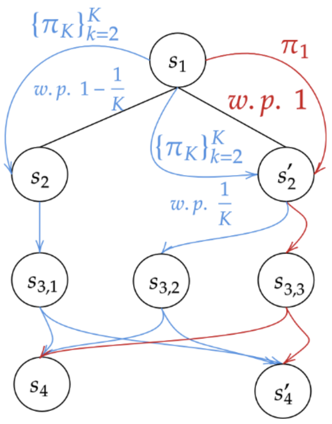

Figure 4: Unfavorable instance for Theorem 11

Begin by fixing the parameters K, ϵ, δ. The topology of the MDP can be found in Figure 4. In state s_1 policy π_1 takes action a_1 and transitions to state s_2' w.p. 1. All other policies take action a_2 and transition to state s_2 w.p. $1 - 1/K$ and to s_2' w.p. $1 - 1/K$. The transition from state s_2 to $s_{3,1}$ is deterministic. There are two actions in state s_2', with π_1 playing a_2 and transitioning to $s_{3,3}$ w.p. 1. All remaining policies play action a_1 and transition to state $s_{3,2}$ w.p. 1. State $s_{3,1}$ contains a single action which transitions uniformly at random to s_4 or s_4'. State $s_{3,2}$ contains $K-1$ actions, indexed by each of the policies $k \in \{2, 3, \dots, K\}$. Action a_k transitions to state s_4 with probability $\frac{1}{2} + \gamma_k$ and to s_4' w.p. $\frac{1}{2} - \gamma_k$, where $\gamma \in \{\epsilon, -\epsilon\}^K$ is a fixed vector which defines the class of MDPs which we consider. State $s_{3,3}$ also contains a single action a_1 which transitions to s_4 with probability $\frac{1}{2} - \gamma_1$ and with the remaining probability to s_4'. States s_4 and s_4' contain a single action a_1 and the only non-zero reward is $r(s_4, a_1) = 1$. The standard lower bound argument requires the round-robin estimator to visit state $s_{3,3}$ at least $\Omega(\frac{1}{\epsilon^2})$ times in expectation to be able to identify the value of γ_1, or equivalently to estimate the return of π_1 up to ϵ accuracy. Thus the expected sample complexity of any round-robin algorithm will be at least $\frac{K}{\epsilon^2}$. Next, we note that $d^{\max}(s_{3,2}) = \frac{1}{K}$ and that the expected disagreement is $O(1)$, which allows us to evaluate the bound in Theorem 11 as $\tilde{O}\left(\frac{K}{\epsilon^2}\right)$. This bound is in fact tight, since the sampling oracle defined by Algorithm 3 and Algorithm 4 will indeed need $\Omega(K)$ sampled trajectories to reach $s_{3,2}$. Finally, we note that the sampling oracle defined in the proof of Theorem 12 will only need $\tilde{O}(1)$ samples to reach state $s_{3,2}$ as $\mathfrak{d}(s_{3,2}) = 1$. This allows us to evaluate the sample complexity stated in Theorem 12 to $\tilde{O}\left(\frac{1}{\epsilon^2}\right)$.

6 CONCLUSION

We study the multiple-policy high-confidence policy evaluation. Our goal is to devise algorithms which can take favorable structure of the target policy set, such as overlap between the actions that the target policies select. To this extent we propose a new algorithm for the problem (Algorithm 1) which makes use of a sampling oracle and show guarantees in terms of a new measure of similarity which we call *disagreement*. We demonstrate example MDPs in which Algorithm 1 can be more favorable than a naive round-robin MC approach and a model-based estimator, and discuss MDPs in which the algorithm is sub-optimal. Further, we instantiate the algorithm with several different sampling oracles and discuss the sample complexity of the resulting algorithms in Theorem 11 and Theorem 12.

References

Mohammad Gheshlaghi Azar, Ian Osband, and Rémi Munos. Minimax regret bounds for reinforcement learning. In *International Conference on Machine Learning*, pages 263–272. PMLR, 2017.

Bo Dai, Ofir Nachum, Yinlam Chow, Lihong Li, Csaba

Szepesvári, and Dale Schuurmans. Coindice: Off-policy confidence interval estimation. *Advances in neural information processing systems*, 33:9398–9411, 2020.

Christoph Dann, Lihong Li, Wei Wei, and Emma Brunskill. Policy certificates: Towards accountable reinforcement learning. In *International Conference on Machine Learning*, pages 1507–1516, 2019.

Mehrdad Farajtabar, Yinlam Chow, and Mohammad Ghavamzadeh. More robust doubly robust off-policy evaluation. In *International Conference on Machine Learning*, pages 1447–1456. PMLR, 2018.

Yihao Feng, Ziyang Tang, Na Zhang, and Qiang Liu. Non-asymptotic confidence intervals of off-policy evaluation: Primal and dual bounds. *arXiv preprint arXiv:2103.05741*, 2021.

Raphael Fonteneau, Susan A Murphy, Louis Wehenkel, and Damien Ernst. Batch mode reinforcement learning based on the synthesis of artificial trajectories. *Annals of operations research*, 208(1):383–416, 2013.

Omer Gottesman, Yao Liu, Scott Sussex, Emma Brunskill, and Finale Doshi-Velez. Combining parametric and nonparametric models for off-policy evaluation. In *International Conference on Machine Learning*, pages 2366–2375. PMLR, 2019.

Asela Gunawardana, Guy Shani, and Sivan Yogev. Evaluating recommender systems. In *Recommender systems handbook*, pages 547–601. Springer, 2012.

Josiah P Hanna, Peter Stone, and Scott Niekum. Bootstrapping with models: Confidence intervals for off-policy evaluation. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Botao Hao, Xiang Ji, Yaqi Duan, Hao Lu, Csaba Szepesvari, and Mengdi Wang. Bootstrapping fitted q-evaluation for off-policy inference. In *International Conference on Machine Learning*, pages 4074–4084. PMLR, 2021.

Nan Jiang and Jiawei Huang. Minimax value interval for off-policy evaluation and policy optimization. *Advances in Neural Information Processing Systems*, 33: 2747–2758, 2020.

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

Chi Jin, Akshay Krishnamurthy, Max Simchowitz, and Tiancheng Yu. Reward-free exploration for reinforcement learning. In *International Conference on Machine Learning*, pages 4870–4879. PMLR, 2020.

Lihong Li, Rémi Munos, and Csaba Szepesvári. Toward minimax off-policy value estimation. In *Artificial Intelligence and Statistics*, pages 608–616. PMLR, 2015.

Qiang Liu, Lihong Li, Ziyang Tang, and Dengyong Zhou. Breaking the curse of horizon: Infinite-horizon off-policy estimation. *Advances in Neural Information Processing Systems*, 31, 2018.

Yao Liu, Pierre-Luc Bacon, and Emma Brunskill. Understanding the curse of horizon in off-policy evaluation via conditional importance sampling. In *International Conference on Machine Learning*, pages 6184–6193. PMLR, 2020.

Alberto Maria Metelli, Matteo Papini, Nico Montali, and Marcello Restelli. Importance sampling techniques for policy optimization. *The Journal of Machine Learning Research*, 21(1):5552–5626, 2020.

Alberto Maria Metelli, Alessio Russo, and Marcello Restelli. Subgaussian and differentiable importance sampling for off-policy evaluation and learning. *Advances in Neural Information Processing Systems*, 34: 8119–8132, 2021.

Andrew Y Ng and Michael Jordan. Pegasus: a policy search method for large mdps and pomdps. In *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*, pages 406–415, 2000.

Matteo Papini, Alberto Maria Metelli, Lorenzo Lupo, and Marcello Restelli. Optimistic policy optimization via multiple importance sampling. In *International Conference on Machine Learning*, pages 4989–4999. PMLR, 2019.

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

Scott Sussex, Omer Gottesman, Yao Liu, Susan Murphy, Emma Brunskill, and Finale Doshi-Velez. Stitched trajectories for off-policy learning.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction.* MIT press, 1992.

Masatoshi Uehara, Masaaki Imaizumi, Nan Jiang, Nathan Kallus, Wen Sun, and Tengyang Xie. Finite sample analysis of minimax offline reinforcement learning: Completeness, fast rates and first-order efficiency. *arXiv preprint arXiv:2102.02981*, 2021.

Ruosong Wang, Simon S Du, Lin Yang, and Sham Kakade. Is long horizon rl more difficult than short horizon rl? In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 9075–9085. Curran Associates, Inc., 2020a. URL `https://proceedings.neurips.cc/paper/2020/file/6734fa703f6633ab896eecbdfad8953a-Paper.pdf`.

Ruosong Wang, Simon S Du, Lin F Yang, and Sham M Kakade. Is long horizon reinforcement learning more difficult than short horizon reinforcement learning? *arXiv preprint arXiv:2005.00527*, 2020b.

Ming Yin and Yu-Xiang Wang. Asymptotically efficient off-policy evaluation for tabular reinforcement learning. In *International Conference on Artificial Intelligence and Statistics*, pages 3948–3958. PMLR, 2020.

Ming Yin, Yu Bai, and Yu-Xiang Wang. Near-optimal provable uniform convergence in offline policy evaluation for reinforcement learning. In *International Conference on Artificial Intelligence and Statistics*, pages 1567–1575. PMLR, 2021.

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*, pages 7304–7312. PMLR, 2019.

A Proofs from Section 4

Proof of Lemma 8. Consider the MDP in Figure 3. We assume that $\frac{1}{2}\log_2(K)$ is an integer. The MDP has $H = \frac{1}{2}\log_2(K)$ layers. In the first layer there is a single starting state $s_{1,1}$ with a single action a_1 on which all K policies agree. The second layer of the MDP has two states $s_{2,1}$ and $s_{2,2}$. The transition kernel is set as $P(s_{2,1}|s_{1,1}, a_1) = P(s_{2,1}|s_{1,1}, a_1) = \frac{1}{2}$. Both states $s_{2,1}$ and $s_{2,2}$ contain two actions a_1 and a_2. For $k \in [1, K/2]$ we have $\pi_k(s_{2,i}) = a_{2,1}$ for $k \in [K/2+1, K]$ $\pi_k(s_{2,i}) = a_{2,2}$, that is the set of K policies is split in half at layer 2. The transition kernel is defined as

$$P(s_{3,1}|s_{2,1}, a_1) = 1$$
$$P(s_{3,2}|s_{2,1}, a_2) = 1$$
$$P(s_{3,2}|s_{2,2}, a_1) = 1$$
$$P(s_{3,3}|s_{2,2}, a_2) = 1.$$

For the remaining layers the construction repeats in a similar way. States $s_{2h+1,1}$ and $s_{2h+1,3}$ contain a single action which transitions to $s_{2h+2,1}$ and $s_{2h+2,4}$ respectively and similarly $s_{2h+2,1}$ and $s_{2h+2,4}$ contain a single action which transitions to $s_{2h+3,1}$ and $s_{2h+3,3}$ respectively. States $s_{2h+1,2}$ also contain a single action and the transition is to states $s_{2h+2,2}$ and $s_{2h+2,3}$. States $s_{2h+2,2}$ and $s_{2h+2,3}$ behave similarly to $s_{2,1}$ and $s_{2,2}$ by dividing the set of policies into equal halves. Assume that at state $s_{2h+2,2}$ we have an arrangement of policies $\{\pi_k\}_{k=i}^{j}$, then policies $\{\pi_k\}_{k=i}^{j/2}$ play action a_1 and transition to $s_{2h+3,1}$, and policies $\{\pi_k\}_{k=j/2+1}^{j}$ play action a_2 and transition to $s_{2h+3,2}$. Similarly for state $s_{2h+2,2}$, $\{\pi_k\}_{k=i}^{j/2}$ play a_1 and transition to $s_{2h+3,2}$, and $\{\pi_k\}_{k=j/2+1}^{j}$ play a_2 and transition to $s_{2h+3,3}$. We make the following observations: at each state $s_{2h+1,2}$ we have an arrangement of exactly $K/2^h$ policies, further, any two arrangements at $s_{2h+1,2}$ have an empty intersection. This implies that at the final layer, the state $\bar{s} \equiv s_{H,2}$ will always have an arrangement of at least $\Theta(\sqrt{K})$ policies. Further, with this construction, the disagreement at every layer is at most 2. To show the lemma we specify the action set at $\bar{s}$ to have $\Theta(\sqrt{K})$ actions, so that every policy $\{\pi_k\}_{k=i}^{j}$ present in an arrangement at $\bar{s}$ will take a different action. Note that this is always possible, e.g., by mapping $\pi_k(\bar{s}) = a_{k-1+1}$, since each arrangement is disjoint. We now note that the disagreement is at least $\Omega(\sqrt{K})$ and further $\max_{k\in K} d^{\pi_k}(\bar{s}, a) \le \frac{1}{\sqrt{K}}$. For every other state-action pair (s, a) it holds that $\max_{k\in K} d^{\pi_k}(s, a) \le 1$ which further shows that $\sum_{h=1}^{H}\sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}}\max_{k\in[K]} d_h^{\pi_k}(s, a) \le O(\log(K))$. □

Proof of Lemma 7. Let $K \ge 2$ and $\epsilon = O(1/K)$ be arbitrary and consider a class of MDPs of the following form. Each MDP has $H = 3$ layers and states $\mathcal{S}_1 = \{s_1\}, \mathcal{S}_2 = \{s_2, s_2'\}$ and $\mathcal{S}_3 = \{s_3, s_3'\}$. The reward function is zero everywhere except for s_3 where it is 1 for all actions. There are 2 actions a_1, a_2 and we restrict our attention per state to only those actions taken by the target policies $\pi_1, \ldots, \pi_k$. In the initial state, policy π_1 takes action a_1 and transitions deterministically to s_2'. All other policies take action a_2 and transition deterministically to s_2. In state s_2, all policies take the same action a_1 and transition to s_3 and s_3' with equal probability. In state s_2', π_1 takes action a_1 and transitions to s_3 with probability $\frac{1}{2} + \gamma_1\epsilon K$ and to s_3' with the remaining probability. Each MDP in the class is identified by a vector $\gamma \in \{-1, +1\}^K$ that determines the bias in the transitions to the final layer. The return of each target policy is

$$v^{\pi_1} = \frac{1}{2} + \gamma_1\epsilon K,$$
$$v^{\pi_i} = \frac{1}{2}, i \ge 2.$$

Hence, in order to construct confidence intervals of size at most ϵ that each contain the return of a target policy with probability $1-\delta$, the algorithm has to learn the correct γ with probability at least $1-\delta$. The occupancy measure of the uniform mixture policy $\bar{\pi}$ for visiting states s_2, s_2' are $d^{\bar{\pi}}(s_2) = \frac{K-1}{K}$ and $d^{\bar{\pi}}(s_2') = \frac{1}{K}$. Using standard techniques for lower bounds, we can show that for any algorithm that learns γ correctly with constant probability it holds that the number of samples $\mathbb{E}[n(s_2', a_1)] = \Omega\left(\frac{1}{\epsilon^2}\right)$, otherwise the value function of v^{π_1} will not be accurately estimated. On the other hand the uniform mixture policy $\bar{\pi}$ after sampling a dataset of size n will visit (s_2, a_1) for $nd^{\bar{\pi}}(s_2', a_1) = \frac{n}{K}$ times in expectation. This implies that

$$n = \Omega\left(\frac{K}{\epsilon^2}\right).$$

On the other hand the expected disagreement is exactly 2 and comes from the initial state s_1. □

B Proofs from Section 5

B.1 Proofs related to preprocessing in Algorithm 4

In this section, we prove the correctness of Algorithm 4 stated in Lemma 9.

Proof of Lemma 9. We will first show in Lemma 13 that the algorithm terminates in the desired number of iterations and epochs. We then show in Lemma 14 that $\mathcal{G}$ has the desired coverage and finally in Lemma 15 the property on the occupancy measures. Combining those results with a union bound completes the proof. $\square$

Lemma 13. *With probability at least $1-O(\delta)$, Algorithm 4 terminates within $\log_2\left(\frac{80S}{\epsilon}\log\frac{c}{\delta}\right)$ iterations and thus samples at most $O\left(\frac{SK}{\epsilon}\log\frac{c}{\delta}\right)$ trajectories.*

Proof. It suffices to show that with the desired probability the termination condition $\max_k \tau_k \leq \frac{10}{3}\log\frac{c}{\delta}$ must be triggered in iteration $i = \log_2\left(\frac{80S}{\epsilon}\log\frac{c}{\delta}\right)$.

We first show that if the escape probability p_k from $\mathcal{G}$ under each policy π_k is at most $\frac{\epsilon}{10}$, then the algorithm terminates with high probability. Let $\ell = \log(c/\delta)$ and $n = \left\lceil\frac{10\ell}{\epsilon}\right\rceil$ be the size of $\mathcal{H}_k$. Then τ_k can be written as the sum of n i.i.d. Bernoulli random variables with parameter p_k. By a simple application of Bernstein's inequality, we have with probability at least $1-\delta/c$ that

$$\begin{aligned}
\tau_k &\leq p_k \cdot n + \sqrt{2n\ell p_k(1-p_k)} + \frac{2\ell}{3} && \text{(Bernstein's inequality)}\\
&\leq p_k \cdot n + \sqrt{2n\ell p_k} + \frac{2\ell}{3} && (p_k \in [0,1])\\
&\leq \frac{3np_k}{2} + \frac{5\ell}{3} && \text{(AM-GM inequality)}\\
&\leq \left(\frac{33}{2}\cdot\frac{p_k}{\epsilon} + \frac{5}{3}\right)\ell. && (n \leq \tfrac{11\ell}{\epsilon})
\end{aligned}$$

Hence, when $p_k \leq \epsilon \cdot \frac{10\cdot 2}{33\cdot 3} \leq \frac{\epsilon}{10}$, then $\tau_k \leq \frac{10}{3}\ell$. Taking a union bound over k implies that with probability at least $1-O(\delta)$ the algorithm returns in iteration i if $\max_k p_k \leq \frac{\epsilon}{10}$.

We will now show that the occupancy measure of any state $s \notin \mathcal{G}$ cannot be too large for any of the policies. Consider any $s \in \mathcal{S}$ and $m = 2^i$. Since $w_k(s)$ is the sum of m i.i.d. Bernoulli random variables with parameter $d^{\pi_k}(s)$, it holds by Bernstein's inequality with probability at least $1-\delta/c$ that

$$\begin{aligned}
d^{\pi_k}(s)m &\leq w_k(s) + \sqrt{2\ell(1-d^{\pi_k}(s))d^{\pi_k}(s)m} + \frac{2\ell}{3}\\
&\leq w_k(s) + \sqrt{2\ell d^{\pi_k}(s)m} + \frac{2\ell}{3}\\
&\leq w_k(s) + \frac{d^{\pi_k}(s)m}{2} + \frac{5\ell}{3}. && \text{(AM-GM inequality)}
\end{aligned}$$

Rearranging terms implies for states $s \notin \mathcal{G}$

$$\frac{d^{\pi_k}(s)}{4} \leq \frac{w_k(s)}{2m} + \frac{5\ell}{6m} \leq \frac{11\ell}{6m}.$$

After taking the union bound over k and s, we can conclude that $d^{\pi_k}(s) \leq \frac{8\ell}{m}$ for all s and k. Further, since

$$p_k \leq \sum_{s \in \mathcal{S}\setminus\mathcal{G}} d^{\pi_k}(s) \leq \frac{8S\ell}{m}$$

for all $k \in [K]$, we can conclude that the algorithm will terminate with high probability if $m = 2^i \geq \frac{80S\ell}{\epsilon}$.

$\square$

Lemma 14. *If Algorithm 4 returns g within the first c/K iterations, then with probability at least $1 - O(\delta)$, the escape probability of π_k from $\mathcal{G}$ for each $k \in [K]$ is at most ϵ. This means that the probability that an episode sampled according to any π_k contains a state outside of $\mathcal{G}$ is at most ϵ.*

Proof. Consider a fixed iteration i and let p_k represent the probability that an episode sampled with π_k visits some state outside of $\mathcal{G}$. Let $\ell = \log(c/\delta)$ and $n = \frac{10\ell}{\epsilon}$ be number of trajectories in $\mathcal{H}_k$. Then τ_k can be written as the sum of n i.i.d. Bernoulli random variables with parameter p_k. By a simple application of Bernstein's inequality, we have with probability at least $1 - \delta/c$ that

$$\begin{aligned} p \cdot n &\leq \tau + \sqrt{2n\ell p_k(1-p_k)} + \frac{2\ell}{3} && \text{(Bernstein's inequality)} \\ &\leq \tau + \sqrt{2n\ell p_k} + \frac{2\ell}{3} && (p_k \in [0,1]) \\ &\leq \tau + \frac{np_k}{2} + \frac{5\ell}{3} && \text{(AM-GM inequality)} \end{aligned}$$

Hence, by rearranging this chain of inequalities, we get that when the algorithm terminates in iteration i

$$p \leq 2\frac{\tau_k}{n} + \frac{10\ell}{3n} \leq \frac{20\ell}{3n} \leq \frac{10\ell}{n} = \epsilon.$$

Taking a union bound over $k \in [K]$ and $i \in [i_{\max}]$ where $i_{\max} = c/K$ finishes the proof. □

Lemma 15. *If Algorithm 4 returns g within the first $\frac{c}{SK}$ iterations, then with probability at least $1 - O(\delta)$, the following property holds for each $s \in \mathcal{G}$*

$$d^{g(s)}(s) \geq \frac{1}{4} \max_{k \in [K]} \hat{d}_k(s) \geq \frac{1}{36} \max_{k \in [K]} d^{\pi_k}(s).$$

Proof. Consider a fixed i, $k \in [K]$ and $s \in \mathcal{S}$. Denote $\ell = \log \frac{c}{\delta}$. Since $w_k(s)$ is the sum of m i.i.d. Bernoulli random variables with parameter $d^{\pi_k}(s)$, it holds by Bernstein's inequality with probability at least $1 - \delta/c$ that

$$\begin{aligned} d^{\pi_k}(s)m &\leq w_k(s) + \sqrt{2\ell(1 - d^{\pi_k}(s))d^{\pi_k}(s)m} + \frac{2\ell}{3} \\ &\leq w_k(s) + \sqrt{2\ell d^{\pi_k}(s)m} + \frac{2\ell}{3} \\ &\leq w_k(s) + \frac{d^{\pi_k}(s)m}{2} + \frac{5\ell}{3}. && \text{(AM-GM inequality)} \end{aligned}$$

Rearranging terms implies that

$$\frac{d^{\pi_k}(s)}{4} \leq \frac{w_k(s)}{2m} + \frac{5\ell}{6m} \leq \frac{w_{k'}(s)}{2m} + \frac{5\ell}{6m}$$

where $k' \in \operatorname{argmax}_k w_k(s)$. Assume $s \in \mathcal{G}$, then $w_{k'}(s) \geq 2\ell$ and hence

$$\frac{d^{\pi_k}(s)}{4} \leq \frac{w_{k'}(s)}{2m} + \frac{5\ell}{6m} \leq \frac{w_{k'}(s)}{m} = \hat{d}_k(s).$$

Conversely, Bernstein's inequality also gives with probability at least $1 - \delta/c$ that

$$\begin{aligned} w_k(s) &\leq d^{\pi_k}(s)m + \sqrt{2\ell(1 - d^{\pi_k}(s))d^{\pi_k}(s)m} + \frac{2\ell}{3} \\ &\leq d^{\pi_k}(s)m + \sqrt{2\ell d^{\pi_k}(s)m} + \frac{2\ell}{3} \\ &\leq \frac{3d^{\pi_k}(s)m}{2} + \frac{5\ell}{3}. && \text{(AM-GM inequality)} \end{aligned}$$

Rearranging terms gives for k with $w_k(s) \geq 2\ell$ that

$$\frac{\hat{d}_k(s)}{6} = \hat{d}_k(s) - \frac{5}{6}\hat{d}_k(s) \leq \hat{d}_k(s) - \frac{5\ell}{3m} \leq \frac{3}{2}d^{\pi_k}(s).$$

Hence, $\hat{d}_k(s) \leq 9d^{\pi_k}(s)$ for such k. Taking a union bound over k, s and i yields the desired statement.

□

Lemma 16. *With probability at least $1 - O(\delta)$, Algorithm 4 returns a function $g\colon \mathcal{G} \to \Pi$ such that for all $s \in \mathcal{G}$*

$$d^{g(s)}(s) \geq \frac{1}{12960} \cdot \frac{\epsilon}{S}$$

Proof. Consider a fixed i, $k \in [K]$ and $s \in \mathcal{S}$ and let $m = 2^i$. By Bernstein's inequality and AM-GM, we get with probability at least $1 - \delta/c$

$$\begin{aligned} w_k(s) &\leq d^{\pi_k}(s)m + \sqrt{2\ell(1 - d^{\pi_k}(s))d^{\pi_k}(s)m} + \frac{2\ell}{3} \\ &\leq d^{\pi_k}(s)m + \sqrt{2\ell d^{\pi_k}(s)m} + \frac{2\ell}{3} \leq \frac{3d^{\pi_k}(s)m}{2} + \frac{5\ell}{3}. \end{aligned}$$

where $\ell = \log(c/\delta)$ Rearranging and using that for all $s \in \mathcal{G}$ there is a k' with $w_{k'}(s) \geq 2\ell$, we have

$$\max_k d^{\pi_k}(s) \geq \frac{2}{9} \cdot \frac{\ell}{m}$$

Applying Lemma 13 and a union bound over s, k and i gives that with probability at least $1 - O(\delta)$, we have for all $s \in \mathcal{G}$

$$\max_k d^{\pi_k}(s) \geq \frac{2}{9} \cdot \frac{\epsilon}{80S}.$$

Combining this with Lemma 15 yields that

$$d^{g(s)}(s) \geq \frac{1}{12960} \cdot \frac{\epsilon}{S}.$$

□

B.2 Proofs for Oracle Algorithm 3

Lemma 17. *Assume that Algorithm 3 receives a $g\colon \mathcal{G} \to \Pi$ such that the escape probability for all $\pi_1, \dots, \pi_k$ from $\mathcal{G}$ is at most ϵ. Then the bias of the sampling oracle is at most $H\epsilon$.*

Proof. Let π_k be a target policy and define $P'_\pi(S_1, S_2, \dots, S_H)$ as the distribution over (state) trajectories induced by sampling repeatedly with Algorithm 3 with inputs S_h and $A_h = \pi_k(S_h)$ to generate S_{h+1}. Further let P_π be the distribution over (state-) trajectories when sampling directly with π in the MDP. Then the bias of $\mathcal{O}_g$ can be written as

$$\beta_{\mathcal{O}_g} = \max_{k \in [K]} \left| \mathbb{E}_{P_{\pi_k}}\left[\sum_{h=1}^H r(S_h, \pi_k(S_h))\right] - \mathbb{E}_{P'_{\pi_k}}\left[\sum_{h=1}^H r(S_h, \pi_k(S_h))\right] \right|$$

Denote by $Z = \sum_{h=1}^H r(S_h, \pi_k(S_h))$ the return random variable and by $\mathcal{E}$ the event that $S_h \in \mathcal{G}$ for all $h \in [H]$. Note that $s_\perp \notin \mathcal{G}$.

We will first show that for all policies π_k, the probability of $\mathcal{E}$ is identical under P_{π_k} and P'_{π_k} using the Markov-property of the process. We write

$$\begin{aligned} P_{\pi_k}(\mathcal{E}) &= \sum_{s_1, \dots, s_H \in \mathcal{G}^H} P_{\pi_k}(S_1 = s_1, \dots, S_H = s_H) \\ &= \sum_{s_1, \dots, s_H \in \mathcal{G}^H} P_{\pi_k}(S_1 = s_1) \cdot \prod_{h=2}^H P_{\pi_k}(S_h = s_h | S_{h-1} = s_{h-1}) && \text{(Markov property)} \end{aligned}$$

Now, by the construction of the oracle, we have that

$$P_{\pi_k}(S_h = s_h | S_{h-1} = s_{h-1}) = P'_{\pi_k}(S_h = s_h | S_{h-1} = s_{h-1})$$

for all $s_h \in \mathcal{G}$ and $s_{h-1} \in \mathcal{G}$ since the oracle returns a next state sample by first reaching $s_{h-1} \in \mathcal{G}$ in some manner and then drawing s_h by executing the action $\pi_k(s_{h-1})$ in the environment. Hence $P_{\pi_k}(\mathcal{E}) = P'_{\pi_k}(\mathcal{E}_H)$ holds. Further note that we actually have just shown the stronger result that

$$P_{\pi_k}(S_1 = s_1, \ldots, S_H = s_H) = P'_{\pi_k}(S_1 = s_1, \ldots, S_H = s_H)$$

for all $(s_1, \ldots, s_H) \in \mathcal{G}^H$. Therefore, it also holds that $\mathbb{E}_{P_{\pi_k}}[Z|\mathcal{E}] = \mathbb{E}_{P'_{\pi_k}}[Z|\mathcal{E}]$. We can now use these properties to bound the bias as

$$\begin{aligned}
\beta_{\mathcal{O}_g} &= \max_{k\in[K]} \left|\mathbb{E}_{P_{\pi_k}}[Z] - \mathbb{E}_{P'_{\pi_k}}[Z]\right| \\
&= \max_{k\in[K]} \left|\mathbb{E}_{P_{\pi_k}}[Z|\mathcal{E}]\, P_{\pi_k}(\mathcal{E}) + \mathbb{E}_{P_{\pi_k}}\left[Z|\bar{\mathcal{E}}\right] P_{\pi_k}(\bar{\mathcal{E}}) - \mathbb{E}_{P'_{\pi_k}}[Z|\mathcal{E}]\, P'_{\pi_k}(\mathcal{E}) - \mathbb{E}_{P'_{\pi_k}}\left[Z|\bar{\mathcal{E}}\right] P'_{\pi_k}(\bar{\mathcal{E}})\right| \\
&= \max_{k\in[K]} \left|\mathbb{E}_{P_{\pi_k}}\left[Z|\bar{\mathcal{E}}\right] P_{\pi_k}(\bar{\mathcal{E}}) - \mathbb{E}_{P'_{\pi_k}}\left[Z|\bar{\mathcal{E}}\right] P'_{\pi_k}(\bar{\mathcal{E}})\right| \\
&= \max_{k\in[K]} \left|\mathbb{E}_{P_{\pi_k}}\left[Z|\bar{\mathcal{E}}\right] - \mathbb{E}_{P'_{\pi_k}}\left[Z|\bar{\mathcal{E}}\right]\right| P_{\pi_k}(\bar{\mathcal{E}}) && (P_{\pi_k}(\bar{\mathcal{E}}) = P'_{\pi_k}(\bar{\mathcal{E}})) \\
&= H \max_{k\in[K]} P_{\pi_k}(\bar{\mathcal{E}}) && (Z \in [0, H]) \\
&\leq H\epsilon && (\text{escape probability bounded for all } \pi_k \text{ by } \epsilon)
\end{aligned}$$

□

Lemma 18. *Assume Algorithm 3 is run with parameter $g\colon \mathcal{G} \to \Pi$ on a state-action pair (s, a) where with $s \in \mathcal{G}$. Then with probability at least $1 - \delta$, the sampling cost is*

$$\frac{\log(1/\delta)}{d^{g(s)}(s)}$$

Proof. Each episode the oracle has probability $p = d^{g(s)}(s)$ to reach s and terminate. Consider the probability that the total number of episodes τ required by the oracle exceeds a number $n = \frac{1}{p}\log(1/\delta)$

$$P(\tau > n) \leq (1-p)^n \leq \exp\left(-pn\right) = \exp\left(\frac{p}{p}\log(\delta)\right) = \delta.$$

□

B.3 Proof of Theorem 11

Proof of Theorem 11. The correctness follows directly from the guarantee for Algorithm 1 in Theorem 3, the bound on the oracle bias in Lemma 17 and the guarantee for the preprocesing phase in Lemma 9. This also implies that the total number of episode collected in the preprocessing phase with accuracy parameter ϵ/H is

$$O\left(\frac{SHK}{\epsilon}\log\frac{\log(SK/\epsilon)}{\delta}\right)$$

It remains to determine the number of episodes collected in the oracle calls.

To that end, consider a fixed sampling cost $c(s) \in [0, c_{\max}]$ associated with each state and denote by

$$X_i = \sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}} \mathbf{1}\left\{\mathcal{K}_i^{1:H}(s,a) > 0\right\} c(s)$$

the sample cost accrued in the i-th call to the trajectory sampler. Note that $X_i \in [0, c_{\max}KH]$. Applying Bernstein's inequality, we get that the total sample cost is bounded as

$$\begin{aligned}
\sum_{i=1^n} X_i &\leq n\mathbb{E}[X_1] + \sqrt{2n\ell\mathbb{E}[X_1]c_{\max}KH} + \frac{2\ell c_{\max}KH}{3} \\
&\leq \frac{3}{2}n\mathbb{E}[X_1] + \frac{5\ell}{3}c_{\max}KH
\end{aligned}$$

with probability at least $1-O(\delta)$ where $\ell=\log(1/\delta)$. Take a union bound over all $\mathcal{S}$ and let $c\colon \mathcal{S}\to\mathbb{N}$ be a sampling cost of the oracle associated with each s. Then the total sampling cost of Algorithm 1 is

$$\frac{3}{2}n\cdot\mathbb{E}\left[\sum_{(s,a)\in\mathcal{K}^{1:H}} c(s)\right]+\frac{5\ell}{3}c_{\max}KH$$

with probability at least $1-O(\delta)$. Here we use $(s,a)\in\mathcal{K}^{1:H}$ to mean the sum over all state-action pairs with non-empty entry in $\mathcal{K}^{1:H}$ and $\mathcal{K}^{1:H}$ is the random arrangement produced by the coupled trajectory sampler.

To determine the sampling cost, note that the total number of calls to the oracle with each state is at most nK. Therefore, by Lemma 18 the number of episodes during each of those calls is bounded for all $s\in\mathcal{G}$ by

$$c(s):=\frac{\log(nSK/\delta)}{d^{g(s)}(s)}$$

with probability at least $1-O(\delta)$. For all $s\in\mathcal{S}\setminus\mathcal{G}$, we set $c(s)=0$. Note that by Lemma 16, we have that c is uniformly bounded by $c_{\max}=\frac{12960S\log(nSK/\delta)}{\epsilon}$. Combining this with the total sampling cost bound above, we get that the total number of episodes sampled during oracle calls from Algorithm 1 is bounded as

$$\begin{aligned}&O\left(n\mathbb{E}\left[\sum_{(s,a)\in\mathcal{K}^{1:H}}\frac{\log(SHK/\epsilon\delta)}{d^{g(s)}(s)}\right]+\frac{SH^2K}{\epsilon}\log^2\frac{SK}{\epsilon\delta}\right)\\ &=O\left(\frac{H^2}{\epsilon^2}\mathbb{E}\left[\sum_{(s,a)\in\mathcal{K}^{1:H}}\frac{1}{d^{\max}(s)}\right]\log^2\frac{SK}{\epsilon\delta}+\frac{SH^2K}{\epsilon}\log^2\frac{SK}{\epsilon\delta}\right)\end{aligned}$$

Hence, the total number of trajectories are bounded by

$$\tilde{O}\left(\frac{H^2}{\epsilon^2}\mathbb{E}\left[\sum_{(s,a)\in\mathcal{K}^{1:H}}\frac{1}{d^{\max}(s)}\right]+\frac{SH^2K}{\epsilon}\right)$$

□

B.4 Proof of Theorem 12

Proof of Theorem 12. The algorithm combines the set $\mathcal{G}$ discovered in Algorithm 4 together with the reward-free exploration idea in Jin et al. (2020). Let $\mathcal{A}_k$ denote the action space of policy k, that is the restriction of $\mathcal{A}$ to the actions only played by π_k on $\mathcal{G}$. Let $\mathcal{A}_{[K]}=\bigcup_{k\in[K]}\mathcal{A}_k$. We begin by constructing an augmented MDP with statespace $\mathcal{G}\bigcup\{\bar{s}\}$, action space $\mathcal{A}_{[K]}$ and transition kernel $\bar{P}$ defined as follows. $\bar{P}(s'|s,a)=P(s'|s,a)$ for all $(s',s,a)\in\mathcal{G}^2\times\mathcal{A}_{[K]}$ and $\bar{P}(\bar{s}|s,a)=1-\sum_{s'\in\mathcal{G}}P(s'|s,a)$. Note that since there are no states $s\in\mathcal{S}\setminus\mathcal{G}$, $\bar{P}$ is well-defined. Next, suppose that we need to emulate the generative model oracle to reach state $s\in\mathcal{G}$. We instantiate the rewards of the augmented MDP with 0 for all $(s',a), s'\neq s$ and equal to 1 on (s,a) for all $a\in\mathcal{A}_{[K]}$. Then the maximum value function v^* in this MDP has value exactly equal to $\mathfrak{d}(s)$.

Next, we use the ORLC algorithm from Dann et al. (2019). Theorem 4.1 (Dann et al., 2019) guarantees that after T rounds of the ORLC algorithm, the algorithm will have returned at least one policy π_t, together with a certificate for sub-optimality $\epsilon_t=\tilde{O}\left(\sqrt{\frac{SAH\log(1/\delta)}{T}}+\frac{S^2AH^3\log(1/\delta)}{T}\right)$ with probability $1-\delta$, where the $\tilde{O}$ notation hides poly-logarithmic factors in S,A,H,T. That is, we have w.p. $1-\delta$

$$|\mathfrak{d}(s)-v^{\pi_t}|\leq\epsilon_t\leq\tilde{O}\left(\sqrt{\frac{SAH\log(1/\delta)}{T}}+\frac{S^2AH^3\log(1/\delta)}{T}\right).$$

The above bound is not entirely sufficient for our purposes, however, a more careful reading of the proof, indeed shows that we can bound

$$\begin{aligned}&|\mathfrak{d}(s)-v^{\pi_t}|\\ \leq&\tilde{O}\left(\sqrt{\frac{SAHv^{\pi_t}\log(1/\delta)}{T}}+\frac{S^2AH^3\log(1/\delta)}{T}\right).\end{aligned}$$

Using the standard self-bounding trick now implies that after $T = O(S^2AH^3\log(1/(\delta\mathfrak{d}(s)))/\mathfrak{d}(s))$ rounds of ORLC it holds that $d^{\pi_t}(s) \geq \frac{\mathfrak{d}(s)}{2}$. We note that $\mathfrak{d}(s) \geq d^{\max}(s)$ for which we already have a constant approximation and so we can set $T = \tilde{\Theta}\left(\frac{|\mathcal{G}|^2KH^3}{d^{\max}(s)}\right)$ large enough to ensure that with probability $1-\delta$ we always find a π_t with $d^{\pi_t}(s) \geq \frac{\mathfrak{d}(s)}{2}$. It is in fact possible to only use $T = \tilde{\Theta}\left(\frac{|\mathcal{G}|^2KH^3}{\mathfrak{d}(s)}\right)$ by hypothesis testing $\log(1/d^{\max}(s))$ values possible values of $\mathfrak{d}(s)$ on an exponential grid of $[0,1]$ for $\log(1/\delta)$ runs of ORLC, however, we leave this argument for future work.

The above argument implies that after $\tilde{\Theta}\left(\sum_{s\in\mathcal{G}}\frac{|\mathcal{G}|^2KH^3}{d^{\max}(s)}\right)$ iterations of ORLC we have a family of policies which can simulate a generative model oracle to state s within at most $O(\log(1/\delta)/\mathfrak{d}(s))$ trajectories. By Lemma 16, we have $d^{\max}(s) = \tilde{\Omega}(\frac{\epsilon}{HS})$ for all $s \in G(\epsilon)$. We can therefore bound the total number of episode required to identify the sampling policies for all $s \in \mathcal{G}$ as

$$\tilde{O}\left(\sum_{s\in\mathcal{G}}\frac{|\mathcal{G}|^2KH^3}{d^{\max}(s)}\right) = \tilde{O}\left(\frac{S|\mathcal{G}|^3KH^4}{\epsilon}\right).$$

□

B.5 Using preprocesisng phase for occupancy measure estimates

As discussed in Section 3, Yin and Wang (2020) proved that the mean-squared error of the model-based round-robin estimator is bounded as

$$\mathbb{E}[(\hat{v}_k^\pi - v_k^\pi)^2] \leq \frac{HK}{n}\cdot\sum_{h=1}^{H}\mathbb{E}_{\pi_k}\left[\frac{d^{\pi_k}(s_h,a_h)}{\sum_{j=1}^{K}d^{\pi_j}(s_h,a_h)}\right] + o\left(\frac{1}{n}\right).$$

We can use the preprocessing phase of our sampling oracle to estimate the RHS of this bound from empirical data and remove the dependency in its lower order terms depend on the smallest occupancy measure among the visited states. We state this result in terms of a fixed number of samples n:

Theorem 19. *There exists an algorithm that after n samples from the round-robin model-based estimator can guarantee that*

$$\mathbb{E}\big[(\hat{v}_k^\pi - v_k^\pi)^2\big] \lesssim \left(\frac{HK}{n}\cdot\sum_{h=1}^{H}\mathbb{E}_{\pi_k}\left[\frac{d^{\pi_k}(s_h,a_h)}{\sum_{j=1}^{K}d^{\pi_j}(s_h,a_h)}\right]\times\left(1+\sqrt{\frac{KS}{n}}\right)+\frac{K^4H^3S}{n^2}\right). \tag{3}$$

Furthermore, the RHS of the inequality in (3) can be estimated up to multiplicative constants with probability $1-\delta$.

Proof. To show the inequality we use the restricted MDP construction from the proof of Theorem 12. Fix a target policy π and let $\epsilon = \sqrt{\frac{1}{n}}$. We can then show using a similar argument to that in Lemma 17 that the occupancy measure between the truncated and original MDP at level $h \in H$ can deviate at most by $O(h\epsilon)$

$$\left|\sum_{s_h}\bar{d}^\pi(s_h) - d^\pi(s_h)\right| \leq 3h\epsilon.$$

This then implies that $|\bar{v}^\pi(s) - v^\pi(s)| \leq H^2\epsilon$ and so the total bias incurred by using the truncated MDP is at most $O(\epsilon)$. To get the first part of the Theorem we now plug into the Theorem 3.1 (Yin and Wang, 2020) and notice that the minimum occupancy measure of the behavior policy is at least $\frac{\epsilon}{K|G(\epsilon)|}$ on the augmented MDP. The second part of the theorem follows from Lemma 16 and Lemma 15. □