Model-Free Active Exploration in Reinforcement Learning

Alessio Russo
Division of Decision and Control Systems
KTH Royal Institute of Technology
Stockholm, SE

Alexandre Proutiere
Division of Decision and Control Systems
KTH Royal Institute of Technology
Stockholm, SE

Abstract

We study the problem of exploration in Reinforcement Learning and present a novel model-free solution. We adopt an information-theoretical viewpoint and start from the instance-specific lower bound of the number of samples that have to be collected to identify a nearly-optimal policy. Deriving this lower bound along with the optimal exploration strategy entails solving an intricate optimization problem and requires a model of the system. In turn, most existing sample optimal exploration algorithms rely on estimating the model. We derive an approximation of the instance-specific lower bound that only involves quantities that can be inferred using model-free approaches. Leveraging this approximation, we devise an ensemble-based model-free exploration strategy applicable to both tabular and continuous Markov decision processes. Numerical results demonstrate that our strategy is able to identify efficient policies faster than state-of-the-art exploration approaches.

1 Introduction

Efficient exploration remains a major challenge for reinforcement learning (RL) algorithms. Over the last two decades, several exploration strategies have been proposed in the literature, often designed with the aim of minimizing regret. These include model-based approaches such as Posterior Sampling for RL [47](PSRL) and Upper Confidence Bounds for RL [5, 34, 3](UCRL), along with model-free UCB-like methods [28, 71]. Regret minimization is a relevant objective when one cares about the rewards accumulated during the learning phase. Nevertheless, an often more important objective is to devise strategies that explore the environment so as to learn efficient policies using the fewest number of samples [22]. Such an objective, referred to as Best Policy Identification (BPI), has been investigated in simplistic Multi-Armed Bandit problems [22, 30] and more recently in tabular MDPs [37, 38]. For these problems, tight instance-specific sample complexity lower bounds are known, as well as model-based algorithms approaching these limits. However, model-based approaches may be computationally expensive or infeasible to obtain. In this paper, we investigate whether we can adapt the design of these algorithms so that they become model-free and hence more practical.

Inspired by [37, 38], we adopt an information-theoretical approach, and design our algorithms starting from an instance-specific lower bound on the sample complexity of learning a nearly-optimal policy in a Markov decision process (MDP). This lower bound is the value of an optimization problem, referred to as the lower bound problem, whose solution dictates the optimal exploration strategy in an environment. Algorithms designed on this instance-specific lower bound, rather than minimax bounds, result in truly adaptive methods, capable of tailoring their exploration strategy according to the specific MDP's learning difficulty. Our method estimates the solution to the lower bound problem and employs it as our exploration strategy. However, we face two major challenges: (1) the

Code repository: `https://github.com/rssalessio/ModelFreeActiveExplorationRL`

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

lower bound problem is non-convex and often intractable; (2) this lower bound problem depends on the initially unknown MDP. In [38], the authors propose MDP-NAS, a model-based algorithm that explores according to the estimated MDP. They convexify the lower bound problem and explore according to the solution of the resulting simplified problem. However, this latter problem still has a complicated dependency on the MDP. Moreover, extending MDP-NAS to large MDPs is challenging since it requires an estimate of the model, and the capability to perform policy iteration. Additionally, MDP-NAS employs a *forced exploration* technique to ensure that the *parametric* uncertainty (the uncertainty about the true underlying MDP) diminishes over time — a method, as we argue later, that we believe not to be efficient in handling this uncertainty.

We propose an alternative way to approximate the lower bound problem, so that its solution can be learnt via a model-free approach. This solution depends only on the Q-function and the variance of the value function. Both quantities can advantageously be inferred using classical stochastic approximation methods. To handle the parametric uncertainty, we propose an ensemble-based method using a bootstrapping technique. This technique is inspired by posterior sampling and allows us to quantify the uncertainty when estimating the Q-function and the variance of the value function.

Our contributions are as follows: (1) we shed light on the role of the instance-specific quantities needed to drive exploration in uncertain MDPs; (2) we derive an alternate upper bound of the lower bound problem that in turn can be approximated using quantities that can be learned in a model-free manner. We then evaluate the quality of this approximation on various environments: (*i*) a random MDP, (*ii*) the Riverswim environment [60], and (*iii*) the Forked Riverswim environment (a novel environment with high sample complexity); (3) based on this approximation, we present Model Free Best Policy Identification (MF-BPI), a model-free exploration algorithm for tabular and continuous MDPs. For the tabular MDPs, we test the performance of MF-BPI on the Riverswim and the Forked Riverswim environments, and compare it to that of Q-UCB [28, 71], PSRL[47], and MDP-NAS[38]. For continuous state-spaces, we compare our algorithm to IDS[44] and BSP [50] (Boostrapped DQN with randomized prior value functions) and assess their performance on hard-exploration problems from the DeepMind BSuite [52] (the DeepSea and the Cartpole swingup problems).

2 Related Work

The body of work related to exploration methods in RL problems is vast, and we mainly focus on online discounted MDPs (for the generative setting, refer to the analysis presented in [23, 37]). Exploration strategies in RL often draw inspiration from the approaches used in multi-armed bandit problems [35, 62], including ϵ-greedy exploration, Boltzmann exploration [73, 62, 35, 2], or more advanced procedures, such as Upper-Confidence Bounds (UCB) methods [3, 4, 35] or Bayesian procedures [65, 74, 19, 56]. We first discuss tabular MDPs, and then extend the discussion to the case of RL with function approximation.

Exploration in tabular MDPs. Numerous algorithms have been proposed with the aim of matching the PAC sample complexity minimax lower bound $\tilde{\Omega}\left(\frac{|S||A|}{\varepsilon^2(1-\gamma)^3}\right)$ [34]. In the design of these algorithms, model-free approaches typically rely on a UCB-like exploration [3, 35], whereas model-based methods leverage estimates of the MDP to drive the exploration. Some well-known model-free algorithms are MEDIAN-PAC [54], DELAYED Q-LEARNING [61] and Q-UCB [71, 28]. Some notable model-based algorithms include: DEL [45], an algorithm that achieves asymptotically optimal instance-dependent regret; UCRL [34], an algorithm that uses extended value-iteration to compute an optimistic MDP; PSRL [47], that uses posterior sampling to sample an MDP. Other algorithms include MBIE [60], E3 [31], R-MAX [14, 29], and MORMAX [63]. Most of existing algorithms are designed towards regret minimization. Recently, however, there has been a growing interest towards exploration strategies with minimal sample complexity, see e.g. [76, 37]. In [37, 38], the authors showed that computing an exploration strategy with minimal sample complexity requires to solve a non-convex problem. To overcome this challenge, they derived a tractable approximation of the lower bound problem, whose solution provides an efficient exploration policy under the generative model [37] and the forward model [38]. This policy necessitates an estimate of the model, and includes a forced exploration phase (an ϵ-soft policy to guarantee that all state-action pairs are visited infinitely often). In [64], the above procedure is extended to linear MDPs, but there again, computing an optimal exploration strategy remains challenging. On a side note, in [70], the authors provide an

alternative bound in the tabular case for episodic MDPs, and later extend it to linear MDPs [69]. The episodic setting is further explored in [66] for deterministic MDPs.

Exploration in Deep Reinforcement Learning (DRL). Exploration methods in DRL environments face several challenges, related to the fact that the state-action spaces are often continuous, and other issues related to training deep neural architectures [58]. The main issue in these large MDPs is that good exploration becomes extremely hard when either the reward is sparse/delayed or the observations contain distracting features [15, 75]. Numerous heuristics have been proposed to tackle these challenges, such as (1) adding an entropy term to the optimization problem to encourage the policy to be more randomized [42, 24] or (2) injecting noise in the observations/parameters [21, 55]. More generally, exploration techniques generally fall into two categories: *uncertainty-based* and *intrinsic-motivation-based* [75, 33]. Uncertainty-based methods decouple the uncertainty into *parametric* and *aleatoric* uncertainty. Parametric uncertainty [19, 43, 32, 75] quantifies the uncertainty in the parameters of the state-action value. This uncertainty vanishes as the agent explores and learns. The aleatoric uncertainty accounts for the inherent randomness of the environment and of the policy [43, 32, 75]. Various methods have been proposed to address the parametric uncertainty, including UCB-like mechanisms [16, 75], or TS-like (Thompson Sampling) techniques [49, 47, 6, 46, 48, 51]. However, computing a posterior of the Q-values is a difficult task. For instance, Bayesian DQN [6] extends Randomized Least-Squares Value Iteration (RLSVI) [49] by considering the features prior to the output layer of the deep-Q network as a fixed feature vector, in order to recast the problem as a linear MDP. Non-parametric posterior sampling methods include Bootstrapped DQN (and Bootstrapped DQN with prior functions) [48, 50, 51], which maintains several independent Q-value functions and randomly samples one of them to explore the environment. Bootstrapped DQN was extended in various ways by integrating other techniques [7, 36]. For the sake of brevity, we refer the reader to the survey in [75] for an exhaustive list of algorithms. Most of these algorithms do not directly account for aleatoric uncertainty in the value function. This uncertainty is usually estimated using methods like Distributional RL [11, 18, 39]. Well-known exploration methods that account for both aleatoric and epistemic uncertainties include Double Uncertain Value Network (DUVN) [43] and Information Directed Sampling (IDS) [32, 44]. The former uses Bayesian dropout to measure the epistemic uncertainty, and the latter uses distributional RL [11] to estimate the variance of the returns. In addition, IDS uses bootstrapped DQN to estimate the parametric uncertainty in the form of a bound on the estimate of the suboptimality gaps. These uncertainties are then combined to compute an exploration strategy. Similarly, in [17], the authors propose UA-DQN, an approach that uses QR-DQN [18] to learn the parametric and aleatoric uncertainties from the quantile networks. Lastly, we refer the reader to [75, 57, 8] for the class of intrinsic-motivation-based methods.

3 Preliminaries

Markov Decision Process. We consider an infinite-horizon discounted Markov Decision Process (MDP), defined by the tuple $\phi = (S, A, P, q, \gamma, p_0)$. S is the state space, A is the action space, $P : S \times A \to \Delta(S)$ is the distribution over the next state given a state-action pair (s, a), $q : S \times A \to \Delta([0, 1])$ is the distribution of the collected reward (with support in $[0, 1]$), $\gamma \in [0, 1)$ is the discount factor and p_0 is the distribution over the initial state.

Let $\pi : \mathcal{S} \to \Delta(A)$ be a stationary Markovian policy that maps a state to a distribution over actions, and denote by $r(s, a) = \mathbb{E}_{r\sim q(\cdot|s,a)}[r]$ the average reward collected when an action a is chosen in state s. We denote by $V^\pi(s) = \mathbb{E}^\pi_\phi[\sum_{t\geq 0} \gamma^t r(s_t, a_t)|s_0 = s]$ the discounted value of policy π. We denote by $\pi^\star$ an optimal stationary policy: for any $s \in \mathcal{S}$, $\pi^\star(s) \in \arg\max_\pi V^\pi(s)$ and define $V^\star(s) = \max_\pi V^\pi(s)$. For the sake of simplicity, we assume that the MDP has a unique optimal policy (we extend our results to more general MDPs in the appendix). We further define $\Pi^\star_\varepsilon(\phi) = \{\pi : \|V^\pi - V^{\pi^\star}\|_\infty \leq \varepsilon\}$, the set of ε-optimal policies in ϕ for $\varepsilon \geq 0$. Finally, to avoid technicalities, we assume (as in [38]) that the MDP ϕ is communicating (that is, for every pair of states (s, s'), there exists a deterministic policy π such that state s' is accessible from state s using π).

We denote by $Q^\pi(s, a) \coloneqq r(s, a) + \gamma\mathbb{E}_{s'\sim P(\cdot|s,a)}[V^\pi(s')]$ the Q-function of π in state (s, a). We also define the sub-optimality gap of action a in state s to be $\Delta(s, a) \coloneqq Q^\star(s, \pi^\star(s)) - Q^\star(s, a)$, where $Q^\star$ is the Q-function of $\pi^\star$, and let $\Delta_{\min} \coloneqq \min_{s,a\neq\pi^\star(s)} \Delta(s, a)$ be the minimum gap in ϕ. For some policy π, we define $\mathrm{Var}_{sa}[V^\pi] \coloneqq \mathrm{Var}_{s'\sim P(\cdot|s,a)}[V^\pi(s')]$ to be the variance of the value function V^π in the next state after taking action a in state s. More generally, we define $M^k_{sa}[V^\pi] \coloneqq$

$\mathbb{E}_{s'\sim P(\cdot|s,a)}\left[\left(V^\pi(s') - \mathbb{E}_{\bar{s}\sim P(\cdot|s,a)}[V^\pi(\bar{s})]\right)^{2^k}\right]$ to be the 2^k-th moment of the value function in the next state after taking action a in state s. We also let $\mathrm{MD}_{sa}[V^\pi] \coloneqq \|V^\pi - \mathbb{E}_{s'\sim P(\cdot|s,a)}[V^\pi]\|_\infty$ be the span of ϕ under π, *i.e.*, the maximum deviation from the mean of the next state value after taking action a in state s.

Best policy identification and sample complexity lower bounds. The MDP ϕ is initially unknown, and we are interested in the scenario where the agent interacts sequentially with ϕ. In each round $t \in \mathbb{N}$, the agent selects an action a_t and observes the next state and the reward (s_{t+1}, r_t): $s_{t+1} \sim P(\cdot|s_t, a_t)$ and $r_t \sim q(\cdot|s_t, a_t)$. The objective of the agent is to learn a policy in $\Pi^\star_\varepsilon(\phi)$ (possibly $\pi^\star$) as fast as possible. This objective is often formalized in a PAC framework where the learner has to stop interacting with the MDP when she can output an ε-optimal policy with probability at least $1-\delta$. In this formalism, the learner strategy consists of (i) a sampling rule or exploration strategy; (ii) a stopping time τ; (iii) an estimated optimal policy $\hat{\pi}$. The strategy is called (ε, δ)-PAC if it stops almost surely, and $\mathbb{P}_\phi[\hat{\pi} \in \Pi^\star_\varepsilon(\phi)] \geq 1-\delta$. Interestingly, one may derive instance-specific lower bounds of the sample complexity $\mathbb{E}_\phi[\tau]$ of any (ε, δ)-PAC algorithm [37, 38], which involves computing an optimal allocation vector $\omega_{\mathrm{opt}} \in \Delta(S \times A)$ (where $\Delta(S \times A)$ is the set of distributions over $S \times A$) that specifies the proportion of times an agent needs to sample each pair (s, a) to confidently identify the optimal policy:

$$\liminf_{\delta \to 0} \frac{\mathbb{E}_\phi[\tau]}{\mathrm{kl}(\delta, 1-\delta)} \geq T_\varepsilon(\omega_{\mathrm{opt}}) \text{ where } T_\varepsilon(\omega)^{-1} \coloneqq \inf_{\psi \in \mathrm{Alt}_\varepsilon(\phi)} \mathbb{E}_{(s,a)\sim\omega}[\mathrm{KL}_{\phi|\psi}(s,a)], \qquad (1)$$

and $\omega_{\mathrm{opt}} = \arg\inf_{\omega \in \Omega(\phi)} T_\varepsilon(\omega)^{-1}$. Here, $\mathrm{Alt}_\varepsilon(\phi)$ is the set of confusing MDPs ψ such that the ε-optimal policies of ϕ are not ε-optimal in ψ, i.e., $\mathrm{Alt}_\varepsilon(\phi) \coloneqq \{\psi : \phi \ll \psi, \Pi^\star_\varepsilon(\phi) \cap \Pi^\star_\varepsilon(\psi) = \emptyset\}$. In this definition, if the next state and reward distributions under ψ are $P'(s,a)$ and $q'(s,a)$, we write $\phi \ll \psi$ if for all (s,a) the distributions of the next state and of the rewards satisfy $P(s,a) \ll P'(s,a)$ and $q(s,a) \ll q'(s,a)$.We further let $\mathrm{KL}_{\phi|\psi}(s,a) \coloneqq \mathrm{KL}(P(s,a), P'(s,a)) + \mathrm{KL}(q(s,a), q'(s,a))$. $\Omega(\phi)$ is the set of possible allocations; in the generative case it is $\Delta(S \times A)$, while with navigation constraints we have $\Omega(\phi) \coloneqq \{\omega \in \Delta(S \times A) : \omega(s) = \sum_{s',a'} P(s|s',a')\omega(s',a')\}, \forall s \in S\}$, with $\omega(s) \coloneqq \sum_a \omega(s,a)$. Finally, $\mathrm{kl}(a,b)$ is the KL-divergence between two Bernoulli distributions of means a and b.

4 Towards Efficient Exploration Allocations

We aim to extend previous studies on best policy identification to online model-free exploration. In this section, we derive an approximation to the bound proposed in [37], involving quantities learnable via stochastic approximation, thereby enabling the use of model-free approaches.

The optimization problem (1) leading to instance-specific sample complexity lower bounds has an important interpretation [37, 38]. An allocation ω_{opt} corresponds to an exploration strategy with minimal sample complexity. To devise an efficient exploration strategy, one could then think of estimating the MDP ϕ, and solving (1) for this estimated MDP to get an approximation of ω_{opt}. There are two important challenges towards applying this approach:

(i) Estimating the model can be difficult, especially for MDPs with large state and action spaces, and arguably, a model-free method would be preferable.

(ii) The lower bound problem (1) is, in general, non-convex [37, 38].

A simple way to circumvent issue (ii) involves deriving an upper bound of the value of the sample complexity lower bound problem (1). Specifically, one may derive an upper bound $U(\omega)$ of $T_\varepsilon(\omega)$ by convexifying the corresponding optimization problem. The exploration strategy can then be the $\omega^\star$ that achieves the infimum of $U(\omega)$. This approach ensures that we identify an approximately optimal policy, at the cost of *over-exploring* at a rate corresponding to the gap $U(\omega^\star) - T_\varepsilon(\omega_{\mathrm{opt}})$. Note that using a lower bound of $T_\varepsilon(\omega)$ would not guarantee the identification of an optimal policy, since we would explore "less" than required. The aforementioned approach was already used in [37] where the authors derive an explicit upper bound $U_0(\omega)$ of $T_0(\omega)$. We also apply it, but derive an upper bound such that implementing the corresponding allocation $\omega^\star$ can be done in a model-free manner (hence solving the first issue (i)).

4.1 Upper bounds on $T_\varepsilon(\omega)$

The next theorem presents the upper bound derived in [37].

Theorem 4.1 ([37])**.** *Consider a communicating MDP ϕ with a unique optimal policy $\pi^\star$. For all vectors $\omega \in \Delta(S \times A)$,*

$$T_0(\omega) \leq U_0(\omega) \coloneqq \max_{(s,a):a\neq\pi^\star(s)} \frac{H_0(s,a)}{\omega(s,a)} + \max_s \frac{H_0^\star}{\omega(s,\pi^\star(s))}, \tag{2}$$

with

$$\begin{cases} H_0(s,a) = \frac{2}{\Delta(s,a)^2} + \max\left(\frac{16\,\mathrm{Var}_{sa}[V^\star]}{\Delta(s,a)^2}, \frac{6\,\mathrm{MD}_{sa}[V^\star]^{4/3}}{\Delta(s,a)^{\frac{4}{3}}}\right), \\ H_0^\star = \frac{2}{\Delta_{\min}^2(1-\gamma)^2} + \min\left(\frac{27}{\Delta_{\min}^2(1-\gamma)^3}, \max\left(\frac{16\max_s \mathrm{Var}_{s\pi^\star(s)}[V^\star]}{\Delta_{\min}^2(1-\gamma)^2}, \frac{6\max_s \mathrm{MD}_{s\pi^\star(s)}[V^\star]^{4/3}}{\Delta_{\min}^{4/3}(1-\gamma)^{4/3}}\right)\right). \end{cases}$$

In the upper bound presented in this theorem, the following quantities characterize the *hardness* of learning the optimal policy: $\Delta(s,a)$ represents the difficulty of learning that in state s action a is sub-optimal; the variance $\mathrm{Var}_{sa}[V^\star]$ measures the aleatoric uncertainty in future state values; and the span $\mathrm{MD}_{sa}[V^\star]$ of the optimal value function can be seen as another measure of aleatoric uncertainty, large whenever there is a significant variability in the value for the possible next states.

Estimating the span $\mathrm{MD}_{sa}[V^\star]$, in an online setting, is a challenging task for large MDPs. Our objective is to derive an alternative upper bound that, in turn, can be approximated using quantities that can be learned in a model-free manner. We observe that the variance of the value function, and more generally its moments $M_{sa}^k[V^\star]^{2^{-k}}$ for $k \geq 1$ (see Appendix C), are smaller than the span. By refining the proof techniques used in [37], we derive the following alternative upper bound.

Theorem 4.2. *Let $\varepsilon \geq 0$ and let $k(s,a) \coloneqq \arg\sup_{k\in\mathbb{N}} M_{sa}^k[V^\star]^{2^{-k}}$ (for brevity, we write k instead of $k(s,a)$). Then, $\forall \omega \in \Delta(S \times A)$, we have $T_\varepsilon(\omega) \leq U(\omega)$, with*

$$U(\omega) \coloneqq \max_{s,a\neq\pi^\star(s)} \left(\frac{2 + 8\varphi^2 M_{sa}^k[V^\star]^{2^{1-k}}}{\omega(s,a)\Delta(s,a)^2} + \max_{s'} \frac{C(s')(1+\gamma)^2}{\omega(s',\pi^\star(s'))\Delta(s,a)^2(1-\gamma)^2}\right), \tag{3}$$

where $C(s') = \max\left(4, 16\gamma^2\varphi^2 M_{s',\pi^\star(s')}^k[V^\star]^{2^{1-k}}\right)$ and φ is the golden ratio.

We can observe that in the worst case, the upper bound $U(\omega^\star)$ of the sample complexity lower bound, with $\omega^\star = \arg\inf_\omega U(\omega)$, scales as $O(\frac{|S||A|\max_s \mathrm{MD}_{s,\pi^\star(s)}[V^\star]^2}{\Delta_{\min}^2(1-\gamma)^2})$. Since $\mathrm{MD}_{sa}[V^\star] \leq (1-\gamma)^{-1}$, then $U(\omega^\star)$ scales at most as $O(\frac{|S||A|}{\Delta_{\min}^2(1-\gamma)^4})$. However, the following questions arise: (1) Can we select a single value of k that provides a good approximation across all states and actions? (2) How much does this bound improve on that of Theorem 4.1? As we illustrate in the example presented in the next subsection, we believe that actually selecting $k = 1$ for all states and actions leads to sufficiently good results. With this choice, we obtain the following approximation:

$$U_1(\omega) \coloneqq \max_{s,a\neq\pi^\star(s)} \left(\frac{2 + 8\varphi^2\,\mathrm{Var}_{sa}[V^\star]}{\omega(s,a)\Delta(s,a)^2} + \max_{s'} \frac{C'(s')(1+\gamma)^2}{\omega(s',\pi^\star(s'))\Delta(s,a)^2(1-\gamma)^2}\right), \tag{4}$$

where $C'(s') = \max\left(4, 16\gamma^2\varphi^2\,\mathrm{Var}_{s',\pi^\star(s')}[V^\star]\right)$. $U_1(\omega)$ resembles the term in Theorem 4.1 (note that we do not know whether U_1 is a valid upper bound for T_ε). For the second question, our numerical experiments (presented below) suggest that $U(\omega)$ is a tighter upper bound than $U_0(\omega)$.

4.2 Example on Tabular MDPs

In Figure 1, we compare the characteristic time upper bounds obtained in the previous subsection. These upper bounds correspond to the allocations $\omega^\star$, $\omega_0^\star$, and $\omega_1^\star$ obtained by minimizing, over $\Delta(S \times A)$[1], $U(\omega)$, $U_0(\omega)$, and $U_1(\omega)$, respectively. We evaluated these characteristic times on various MDPs: (1) a random MDP (see Appendix A); (2) the `RiverSwim` environment [60]; (3) the

[1]Results are similar when we account for the navigation constraints. We omit these results for simplicity.

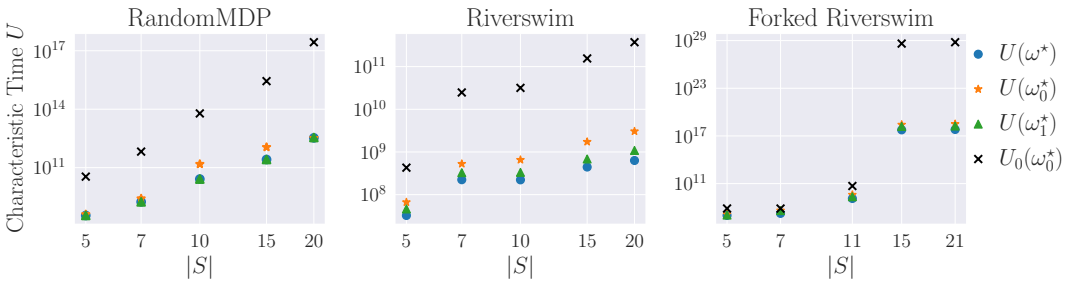

Figure 1: Comparison of the upper bounds (2) and (3) for different sizes of S and $\gamma = 0.95$. We evaluated different allocations using $U_0(\omega)$ and $U(\omega)$. The allocations are: $\omega_0^\star$ (the optimal allocation in (2), $\omega^\star$ (the optimal allocation in (3) and $\omega_1^\star$ (the optimal allocation in (4) by setting $k = 1$ uniformly across states and actions). For the random MDP we show the median value across 30 runs.

`Forked RiverSwim`, a novel environment where the agent needs to constantly explore two different states to learn the optimal policy (compared to the `RiverSwim` environment, the sample complexity is higher; refer to Appendix A for a complete description).

We note that across all plots, the optimal allocation $\omega_0^\star$ has a quite large characteristic time (black cross). Instead, the optimal allocation $\omega^\star$ (blue circle) computed using our new upper bound (3) achieves a lower characteristic time. When we evaluate $\omega_0^\star$ on the new bound (3) (orange star), we observe similar characteristic times.

Finally, to verify that we can indeed choose $k = 1$ uniformly across states and actions, we evaluated the characteristic time $\omega_1^\star$ computed using (4) (green triangle). Our results indicate that the performance is not different from those obtained with $\omega^\star$, suggesting that the quantities of interest (gaps and variances) are enough to learn an efficient exploration allocation. We investigate the choice of k in more detail in Appendix A .

5 Model-Free Active Exploration Algorithms

In this section we present MF-BPI, a model-free exploration algorithm that leverages the optimal allocations obtained through the previously derived upper bound of the sample complexity lower bound. We first present an upper bound $\tilde{U}(\omega)$ of $U(\omega)$, so that it is possible to derive a closed form solution of the optimal allocation (an idea previously proposed in [37]).

Proposition 5.1. *Assume that ϕ has a unique optimal policy $\pi^\star$. For all $\omega \in \Delta(S \times A)$, we have:*

$$U(\omega) \leq \tilde{U}(\omega) := \max_{s,a \neq \pi^\star(s)} \frac{H(s,a)}{\omega(s,a)} + \frac{H}{\min_{s'} \omega(s', \pi^\star(s'))},$$

with $H(s,a) := \frac{2+8\varphi^2 M_{sa}^k [V^\star]^{2^{1-k}}}{\Delta(s,a)^2}$ and $H := \frac{\max_{s'} C(s')(1+\gamma)^2}{\Delta_{\min}^2 (1-\gamma)^2}$. The minimizer $\tilde{\omega}^\star := \arg\inf_\omega \tilde{U}(\omega)$ satisfies $\tilde{\omega}^\star(s,a) \propto H(s,a)$ for $a \neq \pi^\star(s)$ and $\tilde{\omega}^\star(s, \pi^\star(s)) \propto \sqrt{H \sum_{s,a \neq \pi^\star(s)} H(s,a)/|S|}$ otherwise.

In the MF-BPI algorithm, we estimate the gaps $\Delta(s,a)$ and $M_{sa}^k[V^\star]$ for a fixed small value of k (we later explain how to do this in a model-free manner.) and compute the corresponding allocation $\tilde{\omega}^\star$. This allocation drives the exploration under MF-BPI. Using this design approach, we face two issues:

(1) Uniform k and regularization. It is impractical to estimate $M_{sa}^k[V^\star]$ for multiple values of k. Instead, we fix a small value of k (*e.g.*, $k = 1$ or $k = 2$) for all state-action pairs (refer to the previous section for a discussion on this choice). Then, to avoid excessively small values of the gaps in the denominator, we regularize the allocation $\tilde{\omega}^\star$ by replacing, in the expression of $H(s,a)$ (resp. $H_{\min}$), $\Delta(s,a)$ (resp. $\Delta_{\min}$) by $(\Delta(s,a) + \lambda)$ (resp. $(\Delta_{\min} + \lambda)$) for some $\lambda > 0$.

(2) Handling parametric uncertainty via bootstrapping. The quantities $\Delta(s,a)$ and $M_{sa}^k[V^\star]$ required to compute $\tilde{\omega}^\star$ remain unknown during training, and we adopt the Certainty Equivalence principle, substituting the current estimates of these quantities to compute the exploration strategy.

Algorithm 1 Boostrapped MF-BPI (Boostrapped Model Free Best Policy Identification)

Require: Parameters (λ, k, p); ensemble size B; learning rates $\{(\alpha_t, \beta_t)\}_t$.

1: Initialize $Q_{1,b}(s,a) \sim \mathcal{U}([0, 1/(1-\gamma)])$ and $M_{1,b}(s,a) \sim \mathcal{U}([0, 1/(1-\gamma)^{2^k}])$ for all $(s,a) \in S \times A$ and $b \in [B]$.
2: **for** $t = 0, 1, 2, \ldots,$ **do**
3: Bootstrap a sample $(\hat{Q}_t, \hat{M}_t)$ from the ensemble, and compute the allocation $\omega^{(t)}$ using Proposition 5.1. Sample $a_t \sim \omega^{(t)}(s_t, \cdot)$; observe $(r_t, s_{t+1}) \sim q(\cdot|s_t, a_t) \otimes P(\cdot|s_t, a_t)$.
4: **for** $b = 1, \ldots, B$ **do**
5: With probability p, using the experience (s_t, a_t, r_t, s_{t+1}), update $Q_{t,b}$ and $M_{t,b}$ using Equations (5) and (6).
6: **end for**
7: **end for**

By doing so, we are inherently introducing parametric uncertainty into these terms that is not taken into account by the allocation $\tilde{\omega}^\star$. To deal with this uncertainty, the traditional method, as used e.g. in [37, 38]), involves using ϵ-soft exploration policies to guarantee that all state-action pairs are visited infinitely often. This ensures that the estimation errors vanish as time grows large. In practice, we find this type of forced exploration inefficient. In MF-BPI, we opt for a bootstrapping approach to manage parametric uncertainties, which can augment the traditional forced exploration step, leading to more principled exploration.

5.1 Exploration in tabular MDPs.

The pseudo-code of MF-BPI for tabular MDPs is presented in Algorithm 1. In round t, MF-BPI explores the MDP using the allocation $\omega^{(t)}$ estimating $\tilde{\omega}^\star$. To compute this allocation, we use Proposition 5.1 and need (i) the sub-optimality gaps $\Delta(s,a)$, which can be easily derived from the Q-function; (ii) the 2^k-th moment $M^k_{sa}[V^\star]$, which can always be learnt by means of stochastic approximation. In fact, for any Markovian policy π and pair (s,a) we have $M^k_{sa}[V^\pi_\phi] = \frac{1}{\gamma^{2^k}} \mathbb{E}_{s' \sim P(\cdot|s,a)}[\delta^\pi(s,a,s')^{2^k}]$, where $\delta^\pi(s,a,s') = r(s,a) + \gamma \mathbb{E}_{a' \sim \pi(\cdot|s')}[Q^\pi(s',a')] - Q^\pi(s,a)$ is a variant of the TD-error. MF-BPI then uses an asynchronous two-timescale stochastic approximation algorithm to learn $Q^\star$ and $M^k_{sa}[V^\star]$,

$$Q_{t+1}(s_t,a_t) = Q_t(s_t,a_t) + \alpha_t(s_t,a_t)\left(r_t + \gamma \max_a Q_t(s_{t+1},a) - Q_t(s_t,a_t)\right), \tag{5}$$

$$M_{t+1}(s_t,a_t) = M_t(s_t,a_t) + \beta_t(s_t,a_t)\left((\delta'_t/\gamma)^{2^k} - M_t(s_t,a_t)\right), \tag{6}$$

where $\delta'_t = r_t + \gamma \max_a Q_{t+1}(s_{t+1},a) - Q_{t+1}(s_t,a_t)$, and $\{(\alpha_t,\beta_t)\}_{t\geq 0}$ are learning rates satisfying $\sum_{t\geq 0} \alpha_t(s,a) = \sum_{t\geq 0} \beta_t(s,a) = \infty, \sum_{t\geq 0}(\alpha_t(s,a)^2 + \beta_t(s,a)^2) \leq \infty$, and $\frac{\alpha_t(s,a)}{\beta_t(s,a)} \to 0$.

MF-BPI uses bootstrapping to handle parametric uncertainty. We maintain an ensemble of (Q,M)-values, with B members, from which we sample $(\hat{Q}_t, \hat{M}_t)$ at time t. This sample is generated by sampling a uniform random variable $\xi \sim \mathcal{U}([0,1])$ and, for each (s,a) set $\hat{Q}_t(s,a) = \text{Quantile}_\xi(Q_{t,1}(s,a), \ldots, Q_{t,B}(s,a))$ (assuming a linear interpolation). This method is akin to sampling from the parametric uncertainty distribution (we perform the same operation also to compute $\hat{M}_t$). This sample is used to compute the allocation $\omega^{(t)}$ using Proposition 5.1 by setting $\Delta_t(s,a) = \max_{a'} \hat{Q}_t(s,a') - \hat{Q}_t(s,a)$, $\pi^\star_t(s) = \arg\max_a \hat{Q}_t(s,a)$ and $\Delta_{\min,t} = \min_{s,a \neq \pi^\star_t(s)} \Delta_t(s,a)$. Note that, the allocation $\omega^{(t)}$ can be mixed with a uniform policy, to guarantee asymptotic convergence of the estimates. Upon observing an experience, with probability p, MF-BPI updates a member of the ensemble using this new experience. p tunes the rate at which the models are updated, similar to sampling with replacement, speeding up the learning process. Selecting a high value for p compromises the estimation of the parametric uncertainty, whereas choosing a low value may slow down the learning process.

Exploration without bootstrapping? To illustrate the need for our bootstrapping approach, we tried to use the allocation $\omega^{(t)}$ mixed with a uniform allocation. In Figure 2, we show the results on Riverswim-like environments with 5 states. While forced exploration ensures infinite visits to all

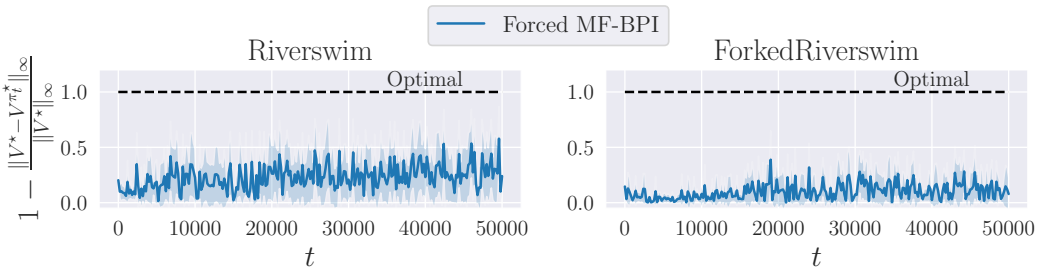

Figure 2: Forced exploration example with 5 states. We explore according to $\omega^{(t)}(s_t, a) = (1 - \epsilon_t)\frac{\tilde{\omega}_t^\star(s_t,a)}{\sum_{a'}\tilde{\omega}_t^\star(s_t,a')} + \epsilon_t \frac{1}{|A|}$, mixing the estimate of the allocation $\tilde{\omega}^\star$ from Proposition 5.1 with a uniform policy, with $\epsilon_t = \max(10^{-3}, 1/N_t(s_t))$ where $N_t(s)$ indicates the number of times the agent visited state s up to time t. Shade indicates 95% confidence interval.

state-action pairs, this guarantee only holds asymptotically. As a result, the allocation mainly focuses on the current MDP estimate, neglecting other plausible MDPs that could produce the same data. This makes the forced exploration approach too sluggish for effective convergence, suggesting its inadequacy for rapid policy learning. These results highlight the need to account for the uncertainty in Q, M when computing the allocation.

5.2 Extension to Deep Reinforcement Learning

To extend bootstrapped MF-BPI to continuous MDPs, we propose DBMF-BPI (see Algorithm 2, or Appendix B). DBMF-BPI uses the mechanism of prior networks from BSP [50](bootstrapping with additive prior) to account for uncertainty that does not originate from the observed data. As before, we keep an ensemble $\{Q_{\theta_1}, \dots, Q_{\theta_B}\}$ of Q-values (with their target networks) and an ensemble $\{M_{\tau_1}, \dots, M_{\tau_B}\}$ of M-values, as well as their prior networks. We use the same procedure as in the tabular case to compute $(\hat{Q}_t, \hat{M}_t)$ at time t, except that we sample $\xi \sim \mathcal{U}([0, 1])$ every $T_s \propto (1-\gamma)^{-1}$ training steps (or at the end of an episode) to make the training procedure more stable. The quantity $\hat{Q}_t$ is used to compute $\pi_t^\star(s_t)$ and $\Delta_t(s_t, a)$. We estimate $\Delta_{\min,t}$ via stochastic approximation, with the minimum gap from the last batch of transitions sampled from the replay buffer serving as a target. To derive the exploration strategy, we compute $H_t(s_t, a) = \frac{2+8\varphi^2 \hat{M}_t(s_t,a)^{2^{1-k}}}{(\Delta_t(s_t,a)+\lambda)^2}$ and $H_t = \frac{4(1+\gamma)^2 \max(1, 4\gamma^2\varphi^2 \hat{M}_t(s_t,\pi_t^\star(s_t))^{2^{1-k}})}{(\Delta_{\min,t}+\lambda)^2(1-\gamma)^2}$. Next, we set the allocation $\omega_o^{(t)}$ as follows: $\omega_o^{(t)}(s_t, a) = H_t(s_t, a)$ if $a \neq \pi_t^\star(s_t)$ and $\omega_o^{(t)}(s_t, a) = \sqrt{H_t \sum_{a \neq \pi_t^\star(s_t)} H_t(s_t, a)}$ otherwise. Finally, we obtain an ϵ_t-soft exploration policy $\omega^{(t)}(s_t, \cdot)$ by mixing $\omega_o^{(t)}(s_t, \cdot)/\sum_a \omega_o^{(t)}(s_t, a)$ with a uniform distribution (using an exploration parameter ϵ_t).

Algorithm 2 DBMF-BPI (Deep Bootstrapped Model Free BPI)

Require: Parameters (λ, k); ensemble size B; exploration rate $\{\epsilon_t\}_t$; estimate $\Delta_{\min,0}$; mask probability p.
1: Initialize replay buffer $\mathcal{D}$, networks Q_{θ_b}, M_{τ_b} and targets $Q_{\theta'_b}$ for all $b \in [B]$.
2: **for** $t = 0, 1, 2, \dots,$ **do**
3: **Sampling step.**
4: Compute allocation $\omega^{(t)} \leftarrow \texttt{ComputeAllocation}(s_t, \{Q_{\theta_b}, M_{\tau_b}\}_{b\in[B]}, \Delta_{\min,t}, \gamma, \lambda, k, \epsilon_t)$.
5: Sample $a_t \sim \omega^{(t)}(s_t, \cdot)$ and observe $(r_t, s_{t+1}) \sim q(\cdot|s_t, a_t) \otimes P(\cdot|s_t, a_t)$.
6: Add transition $z_t = (s_t, a_t, r_t, s_{t+1})$ to the replay buffer $\mathcal{D}$.
7: **Training step.**
8: Sample a batch $\mathcal{B}$ from $\mathcal{D}$, and with probability p add the i^{th} experience in $\mathcal{B}$ to a sub-batch $\mathcal{B}_b$, $\forall b \in [B]$. Update the (Q, M)-values of the b^{th} member in the ensemble using $\mathcal{B}_b$: $\{Q_{\theta_b}, Q_{\theta'_b}, M_{\tau_b}\}_{b\in[B]} \leftarrow \texttt{Training}(\{\mathcal{B}_b, Q_{\theta_b}, Q_{\theta'_b}, M_{\tau_b}\}_{b\in[B]})$.
9: Update estimate $\Delta_{\min,t+1} \leftarrow \texttt{EstimateMinimumGap}(\Delta_{\min,t}, \mathcal{B}, \{Q_{\theta_b}\}_{b\in[B]})$.
10: **end for**

6 Numerical Results

We evaluate the performance of MF-BPI on benchmark problems and compare it against state-of-the-art methods (details can be found in Appendix A).

Tabular MDPs. In the tabular case, we compared various algorithms on the `Riverswim` and `Forked Riverswim` environments. We evaluate MF-BPI with (1) bootstrapping and with (2) the forced exploration step using an ϵ-soft exploration policy, MDP-NAS [38], PSRL [47] and Q-UCB [28, 71]. For MDP-NAS, the model of the MDP was initialized in an optimistic way (with additive smoothing). In both environments, we varied the size of the state space. In Figure 3, we show $1 - \frac{\|V^\star - V^{\pi_T^\star}\|_\infty}{\|V^\star\|_\infty}$, a performance measure for the estimated policy $\pi_T^\star$ after $T = |S| \times 10^4$ steps with $\gamma = 0.99$. Results (the higher the better) indicate that bootstrapped MF-BPI can compete with model-based and model-free algorithms on hard-exploration problems, without resorting to expensive model-based procedures. Details of the experiments, including the initialization of the algorithms, are provided in Appendix A .

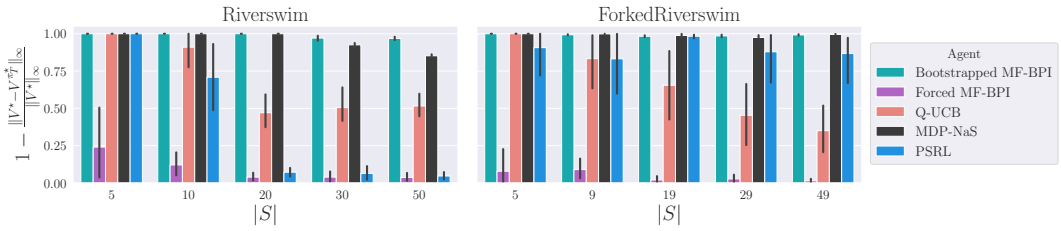

Figure 3: Evaluation of the estimated optimal policy $\pi_T^\star$ after T steps for MF-BPI, Q-UCB, MDP-NAS and PSRL. Results are averaged across 10 seeds and lines indicate 95% confidence intervals.

Deep RL. In environments with continuous state space, we compared DBMF-BPI with BSP [51, 50] (Bootstrapped DQN with randomized priors) and IDS [44] (Information-Directed Sampling). We also evaluated DBMF-BPI against BSP2, a variant of BSP that uses the same masking mechanism as DBMF-BPI for updating the ensemble. These methods were tested on challenging exploration problems from the DeepMind behavior suite [52] with varying levels of difficulty: (1) a stochastic version of DeepSea and (2) the Cartpole swingup problem. The DeepSea problem includes a 5% probability of the agent slipping, i.e., that an incorrect action is executed, which increases the aleatoric variance.

The results for the Cartpole swingup problem are depicted in Figure 4 for various difficulty levels k (see also Appendix A.5 for more details), demonstrating the ability of DBMF-BPI to quickly learn an efficient policy. While BSP generally performs well, there is a notable difference in performance when compared to DBMF-BPI. For a fair comparison, we used the same network initialization across all methods, except for IDS. Untuned, IDS performed poorly; proper initialization improved its performance, but results remained unsatisfactory. In Figure 5, we present two exploration metrics

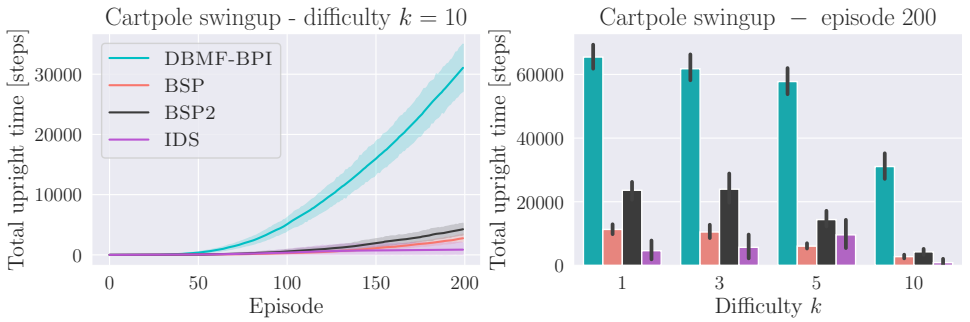

Figure 4: Cartpole swingup problem. On the left: total upright time at a difficulty level of $k = 10$. On the right: total upright time after 200 episodes for different difficulties k. To observe a positive reward, the pole's angle must satisfy $\cos(\theta) > k/20$, and the cart's position should satisfy $|x| \leq 1 - k/20$. Bars and shaded areas indicate 95% confidence intervals.

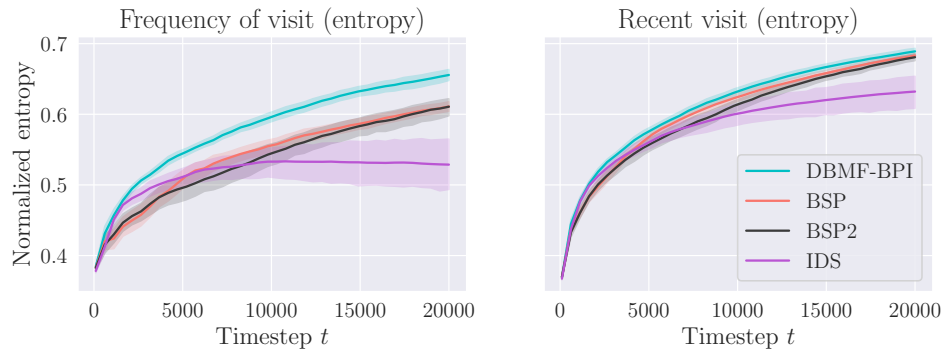

Figure 5: Exploration in Cartpole swingup for $k = 5$. On the left, we show the entropy of visitation frequency for the state space $(x, \dot{x}, \theta, \dot{\theta})$ during training. On the right, we show a measure of the dispersion of the most recent visits; smaller values indicate that the agent is less explorative as t increases.

for difficulty $k = 5$. The frequency of visits measures the uniformity and dispersion of visits across the state space, while the second metric evaluates the recency of visits to different regions, capturing how frequently the methods keep visiting previously visited states (a smaller value indicates that the agent tends to concentrate on a specific region of the state space). For detailed analysis, please refer to appendix A.

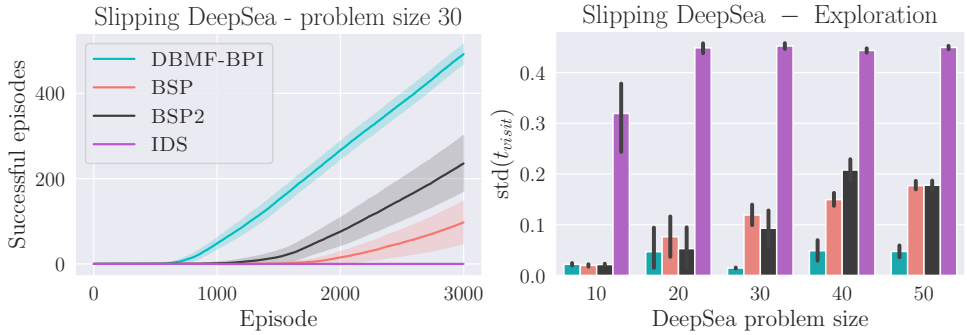

Figure 6: Slipping DeepSea problem. On the left: total number of successful episodes (*i.e.*, that the agent managed to reach the final reward) for a grid with 30^2 input features. On the right: standard deviation of t_{visit} at the last episode, depicting how much each agent explored (the lower the better).

For the slipping DeepSea problem, results are depicted in Fig. 6 (see also Appendix A.4 for more details). Besides the number of successful episodes, we also display the standard deviation of $(t_{\text{visit}})_{ij}$ across all cells (i, j), where $(t_{\text{visit}})_{ij}$ indicates the last timestep t that a cell (i, j) was visited (normalized by NT, the product of the grid size, and the number of episodes). The right plot shows $\text{std}(t_{\text{visit}})$ for different problem sizes, highlighting the good exploration properties of DBMF-BPI. Additional details and exploration metrics can be found in Appendix A .

7 Conclusions

In this work, we studied the problem of exploration in Reinforcement Learning and presented MF-BPI, a model-free solution for both tabular and continuous state-space MDPs. To derive this method, we established a novel approximation of the instance-specific lower bound necessary for identifying nearly-optimal policies. Importantly, this approximation depends only on quantities learnable via stochastic approximation, paving the way towards model-free methods. Numerical results on hard-exploration problems highlighted the effectiveness of our approach for learning efficient policies over state-of-the-art methods.

Acknowledgments

This research was supported by the Swedish Foundation for Strategic Research through the CLAS project (grant RIT17-0046) and partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation. The authors would also like to thank the anonymous reviewers for their valuable and insightful feedback. On a personal note, Alessio Russo wishes to personally thank Damianos Tranos, Yassir Jedra, Daniele Foffano, and Letizia Orsini for their invaluable assistance in reviewing the manuscript.

References

[1] MOSEK ApS. *MOSEK Optimizer API for Python 9.2.49*, 2022. URL `https://docs.mosek.com/9.2/pythonapi/index.html`.

[2] Peter Atkins, Peter William Atkins, and Julio de Paula. *Atkins' physical chemistry*. Oxford university press, 2014.

[3] Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3(Nov):397–422, 2002.

[4] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47:235–256, 2002.

[5] Peter Auer, Thomas Jaksch, and Ronald Ortner. Near-optimal regret bounds for reinforcement learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 21, 2008.

[6] Kamyar Azizzadenesheli, Emma Brunskill, and Animashree Anandkumar. Efficient exploration through bayesian deep q-networks. In *2018 Information Theory and Applications Workshop (ITA)*, pages 1–9. IEEE, 2018.

[7] Chenjia Bai, Lingxiao Wang, Lei Han, Jianye Hao, Animesh Garg, Peng Liu, and Zhaoran Wang. Principled exploration via optimistic bootstrapping and backward induction. In *International Conference on Machine Learning*, pages 577–587. PMLR, 2021.

[8] Andrew G Barto. Intrinsic motivation and reinforcement learning. *Intrinsically Motivated Learning in Natural and Artificial Systems, Springer*, pages 17–47, 2013.

[9] Andrew G. Barto, Richard S. Sutton, and Charles W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13(5):834–846, 1983.

[10] Stefan Behnel, Robert Bradshaw, Craig Citro, Lisandro Dalcin, Dag Sverre Seljebotn, and Kurt Smith. Cython: The best of both worlds. *Computing in Science & Engineering*, 13(2):31–39, 2011.

[11] Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *International Conference on Machine Learning*, pages 449–458. PMLR, 2017.

[12] Rajendra Bhatia and Chandler Davis. A better bound on the variance. *The American Mathematical Monthly*, 107(4):353–357, 2000.

[13] Vivek S Borkar. *Stochastic approximation: a dynamical systems viewpoint*, volume 48. Springer, 2009.

[14] Ronen I Brafman and Moshe Tennenholtz. R-max-a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3(Oct):213–231, 2002.

[15] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. In *International Conference on Learning Representations*, 2018.

[16] Richard Y Chen, Szymon Sidor, Pieter Abbeel, and John Schulman. Ucb exploration via q-ensembles. *arXiv preprint arXiv:1706.01502*, 2017.

[17] William R Clements, Bastien Van Delft, Benoît-Marie Robaglia, Reda Bahi Slaoui, and Sébastien Toth. Estimating risk and uncertainty in deep reinforcement learning. *arXiv preprint arXiv:1905.09638*, 2019.

[18] Will Dabney, Mark Rowland, Marc Bellemare, and Rémi Munos. Distributional reinforcement learning with quantile regression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[19] Richard Dearden, Nir Friedman, and Stuart Russell. Bayesian q-learning. In *Proceedings of the 15th National Conference on Artificial Intelligence, American Association for Artificial Intelligence, July, 1998*, pages 761–768. AAAI Press, 1998.

[20] Steven Diamond and Stephen Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

[21] Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Ian Osband, Alex Graves, Vlad Mnih, Remi Munos, Demis Hassabis, Olivier Pietquin, et al. Noisy networks for exploration. *arXiv preprint arXiv:1706.10295*, 2017.

[22] Aurélien Garivier and Emilie Kaufmann. Optimal best arm identification with fixed confidence. In *Conference on Learning Theory*, pages 998–1027. PMLR, 2016.

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

[24] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pages 1861–1870. PMLR, 2018.

[25] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. doi: 10.1038/s41586-020-2649-2. URL `https://doi.org/10.1038/s41586-020-2649-2`.

[26] Aaron Herschfeld. On infinite radicals. *The American Mathematical Monthly*, 42(7):419–429, 1935.

[27] John D Hunter. Matplotlib: A 2d graphics environment. *Computing in science & engineering*, 9(3):90–95, 2007.

[28] Chi Jin, Zeyuan Allen-Zhu, Sebastien Bubeck, and Michael I Jordan. Is q-learning provably efficient? *Advances in Neural Information Processing Systems (NeurIPS)*, 31, 2018.

[29] Sham Machandranath Kakade. *On the sample complexity of reinforcement learning*. University of London, University College London (United Kingdom), 2003.

[30] Emilie Kaufmann, Olivier Cappé, and Aurélien Garivier. On the complexity of best arm identification in multi-armed bandit models. *Journal of Machine Learning Research*, 17:1–42, 2016.

[31] Michael Kearns and Satinder Singh. Near-optimal reinforcement learning in polynomial time. *Machine learning, Springer*, 49:209–232, 2002.

[32] Johannes Kirschner and Andreas Krause. Information directed sampling and bandits with heteroscedastic noise. In *Conference On Learning Theory*, pages 358–384. PMLR, 2018.

[33] Pawel Ladosz, Lilian Weng, Minwoo Kim, and Hyondong Oh. Exploration in deep reinforcement learning: A survey. *Information Fusion*, 2022.

[34] Tor Lattimore and Marcus Hutter. Pac bounds for discounted mdps. In *Algorithmic Learning Theory: 23rd International Conference, ALT 2012, Lyon, France, October 29-31, 2012. Proceedings 23*, pages 320–334. Springer, 2012.

[35] Tor Lattimore and Csaba Szepesvári. *Bandit algorithms*. Cambridge University Press, 2020.

[36] Kimin Lee, Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Sunrise: A simple unified framework for ensemble learning in deep reinforcement learning. In *International Conference on Machine Learning*, pages 6131–6141. PMLR, 2021.

[37] Aymen Al Marjani and Alexandre Proutiere. Adaptive sampling for best policy identification in markov decision processes. In *International Conference on Machine Learning*, pages 7459–7468. PMLR, 2021.

[38] Aymen Al Marjani, Aurélien Garivier, and Alexandre Proutiere. Navigating to the best policy in markov decision processes. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[39] Borislav Mavrin, Hengshuai Yao, Linglong Kong, Kaiwen Wu, and Yaoliang Yu. Distributional reinforcement learning for efficient exploration. In *International Conference on Machine Learning*, pages 4424–4434. PMLR, 2019.

[40] Wes McKinney et al. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56. Austin, TX, 2010.

[41] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[42] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pages 1928–1937. PMLR, 2016.

[43] Thomas M Moerland, Joost Broekens, and Catholijn M Jonker. Efficient exploration with double uncertain value networks. In *Deep Reinforcement Learning Symposium, NIPS 2017*, 2017.

[44] Nikolay Nikolov, Johannes Kirschner, Felix Berkenkamp, and Andreas Krause. Information-directed exploration for deep reinforcement learning. In *International Conference on Learning Representations*, 2018.

[45] Jungseul Ok, Alexandre Proutiere, and Damianos Tranos. Exploration in structured reinforcement learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 31, 2018.

[46] Ian Osband and Benjamin Van Roy. Bootstrapped thompson sampling and deep exploration. *arXiv preprint arXiv:1507.00300*, 2015.

[47] Ian Osband, Daniel Russo, and Benjamin Van Roy. (more) efficient reinforcement learning via posterior sampling. *Advances in Neural Information Processing Systems (NeurIPS)*, 26, 2013.

[48] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. *Advances in Neural Information Processing Systems (NeurIPS)*, 29, 2016.

[49] Ian Osband, Benjamin Van Roy, and Zheng Wen. Generalization and exploration via randomized value functions. In *International Conference on Machine Learning*, pages 2377–2386. PMLR, 2016.

[50] Ian Osband, John Aslanides, and Albin Cassirer. Randomized prior functions for deep reinforcement learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 31, 2018.

[51] Ian Osband, Benjamin Van Roy, Daniel J Russo, and Zheng Wen. Deep exploration via randomized value functions. *Journal of Machine Learning Research*, 20:1–62, 2019.

[52] Ian Osband, Yotam Doron, Matteo Hessel, John Aslanides, Eren Sezener, Andre Saraiva, Katrina McKinney, Tor Lattimore, Csaba Szepesvári, Satinder Singh, Benjamin Van Roy, Richard Sutton, David Silver, and Hado van Hasselt. Behaviour suite for reinforcement learning. In *International Conference on Learning Representations*, 2020.

[53] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems (NeurIPS)*, 32, 2019.

[54] Jason Pazis, Ronald E Parr, and Jonathan P How. Improving pac exploration using the median of means. *Advances in Neural Information Processing Systems (NeurIPS)*, 29, 2016.

[55] Matthias Plappert, Rein Houthooft, Prafulla Dhariwal, Szymon Sidor, Richard Y Chen, Xi Chen, Tamim Asfour, Pieter Abbeel, and Marcin Andrychowicz. Parameter space noise for exploration. In *International Conference on Learning Representations*, 2018.

[56] Daniel J Russo, Benjamin Van Roy, Abbas Kazerouni, Ian Osband, Zheng Wen, et al. A tutorial on thompson sampling. *Foundations and Trends® in Machine Learning*, 11(1):1–96, 2018.

[57] Richard M Ryan and Edward L Deci. Intrinsic and extrinsic motivations: Classic definitions and new directions. *Contemporary educational psychology*, 25(1):54–67, 2000.

[58] Mohit Sewak. *Deep reinforcement learning*. Springer, 2019.

[59] Satinder Singh, Tommi Jaakkola, Michael L Littman, and Csaba Szepesvári. Convergence results for single-step on-policy reinforcement-learning algorithms. *Machine learning*, 38: 287–308, 2000.

[60] Alexander L Strehl and Michael L Littman. An analysis of model-based interval estimation for markov decision processes. *Journal of Computer and System Sciences*, 74(8):1309–1331, 2008.

[61] Alexander L Strehl, Lihong Li, Eric Wiewiora, John Langford, and Michael L Littman. Pac model-free reinforcement learning. In *Proceedings of the 23rd International Conference on Machine learning*, pages 881–888, 2006.

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

[63] István Szita and Csaba Szepesvári. Model-based reinforcement learning with nearly tight exploration complexity bounds. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 1031–1038, 2010.

[64] Jérôme Taupin, Yassir Jedra, and Alexandre Proutiere. Best policy identification in discounted linear mdps. In *Sixteenth European Workshop on Reinforcement Learning*, 2023.

[65] William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3-4):285–294, 1933.

[66] Andrea Tirinzoni, Aymen Al Marjani, and Emilie Kaufmann. Near instance-optimal pac reinforcement learning for deterministic mdps. *Advances in Neural Information Processing Systems (NeurIPS)*, 35:8785–8798, 2022.

[67] Guido Van Rossum and Fred L Drake Jr. *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam, 1995.

[68] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.

[69] Andrew Wagenmaker and Kevin G Jamieson. Instance-dependent near-optimal policy identification in linear mdps via online experiment design. *Advances in Neural Information Processing Systems (NeurIPS)*, 35:5968–5981, 2022.

[70] Andrew J Wagenmaker, Max Simchowitz, and Kevin Jamieson. Beyond no regret: Instance-dependent pac reinforcement learning. In *Conference on Learning Theory*, pages 358–418. PMLR, 2022.

[71] Yuanhao Wang, Kefan Dong, Xiaoyu Chen, and Liwei Wang. Q-learning with ucb exploration is sample efficient for infinite-horizon mdp. In *International Conference on Learning Representations*, 2019.

[72] Michael Waskom, Olga Botvinnik, Drew O'Kane, Paul Hobson, Saulius Lukauskas, David C Gemperline, Tom Augspurger, Yaroslav Halchenko, John B. Cole, Jordi Warmenhoven, Julian de Ruiter, Cameron Pye, Stephan Hoyer, Jake Vanderplas, Santi Villalba, Gero Kunter, Eric Quintero, Pete Bachant, Marcel Martin, Kyle Meyer, Alistair Miles, Yoav Ram, Tal Yarkoni, Mike Lee Williams, Constantine Evans, Clark Fitzgerald, Brian, Chris Fonnesbeck, Antony Lee, and Adel Qalieh. mwaskom/seaborn: v0.8.1 (september 2017), September 2017.

[73] Christopher John Cornish Hellaby Watkins. Learning from delayed rewards. *PhD thesis, Cambridge University, Cambridge, England*, 1989.

[74] Jeremy Wyatt. Exploration and inference in learning from reinforcement. *PhD thesis, University of Edinburgh, Edinburgh, England*, 1998.

[75] Tianpei Yang, Hongyao Tang, Chenjia Bai, Jinyi Liu, Jianye Hao, Zhaopeng Meng, Peng Liu, and Zhen Wang. Exploration in deep reinforcement learning: a comprehensive survey. *arXiv preprint arXiv:2109.06668*, 2021.

[76] Andrea Zanette, Mykel J Kochenderfer, and Emma Brunskill. Almost horizon-free structure-aware best policy identification with a generative model. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019.

Appendix

Contents

1 Introduction **1**

2 Related Work **2**

3 Preliminaries **3**

4 Towards Efficient Exploration Allocations **4**

4.1 Upper bounds on $T_\varepsilon(\omega)$ 5

4.2 Example on Tabular MDPs 5

5 Model-Free Active Exploration Algorithms **6**

5.1 Exploration in tabular MDPs. 7

5.2 Extension to Deep Reinforcement Learning 8

6 Numerical Results **9**

7 Conclusions **10**

A Numerical Results **19**

A.1 The Forked Riverswim Environment 19

A.2 Details of Example 4.2 19

A.3 Riverswim and Forked Riverswim - Description and Additional Results 22

A.4 Slipping DeepSea - Description and Additional Results 24

A.5 Cartpole Swingup - Description and Additional Results 27

A.6 Parameters, Hardware, Code and Libraries 32

A.6.1 Simulation parameters - Riverswim and Forked Riverswim 32

A.6.2 Simulation parameters - Slipping DeepSea 32

A.6.3 Simulation parameters - Cartpole Swingup 33

A.6.4 Hardware and simulation time 33

A.6.5 Code and libraries 33

B Algorithms **34**

B.1 PS-MDP-NAS - Posterior Sampling for navigating MDPs 34

B.2 O-BPI - Online Best Policy Identification 35

B.3 Boostrapped MF-BPI - Model Free Best Policy Identification 36

B.4 DBMF-BPI - Deep Boostrapped Model Free Best Policy Identification 38

C Proofs **41**

C.1 Preliminaries 41

C.2 Alternative upper bounds 41
C.2.1 Sample complexity lower bound 41
C.2.2 Upper bound on $T_\varepsilon(\omega)$ 42
C.2.3 Closed form solution under the generative model 44
C.2.4 Technical lemmas 45
C.2.5 Decomposition of the set of confusing MDPs 47

Appendix introduction

We start by examining the wider impact of our work and acknowledging its limitations. This provides a balanced view of our contribution and points out areas for future research.

Next, we turn to the numerical results. Here, we give a more detailed account of our findings and include additional results for further clarity. We also introduce and describe the new Forked RiverSwim environment, an advanced version of the existing RiverSwim model, which has a larger sample complexity.

In the subsequent section, we break down the algorithms used in our study. This gives a deeper understanding of the methods underpinning our research.

We wrap up the appendix by providing all the proofs that support our conclusions.

Broader impact

This paper primarily focuses on foundational research in reinforcement learning, specifically the exploration problem, and proposes a novel model-free exploration strategy. While our work does not directly engage with societal impact considerations, we acknowledge the importance of considering the broader implications of AI technologies. As our proposed method improves the efficiency of reinforcement learning algorithms, it could potentially be applied in a wide range of contexts, some of which could have societal impacts. For instance, reinforcement learning is used in decision-making systems, which could include areas like healthcare, finance, and autonomous vehicles, where biases or errors could have significant consequences. Hence, while the direct societal impact of our work may not be immediately apparent, we strongly encourage future researchers and practitioners who apply these techniques to carefully consider the ethical implications and potential negative impacts in their specific use-cases. The responsible use of AI, including the mitigation of bias and the respect for privacy, should always be a priority.

Limitations

While our work presents significant advancements in the area of reinforcement learning, it also has its limitations that need to be acknowledged:

- **Assumptions**: Our approach relies on the assumption that the MDP is communicating. The instance-specific lower bound we propose may not be as effective if this assumption does not hold.
- **Scalability**: Our method, despite being model-free, still relies on stochastic approximations, which may not scale well with the complexity and size of certain MDPs.
- **Comparison with Model-Based Approaches**: While we have shown that our approach performs competitively with existing model-based exploration algorithms in hard-exploration environments, a comprehensive comparison across a wider range of environments is needed. It is possible that our method may not perform as well in some MDPs as the model-based approaches.
- **Bootstrapping**: Although bootstrapping has proven to be an effective technique, its usage is yet to be fully understood in RL applications. To achieve a more profound theoretical comprehension, a comprehensive analysis is necessary.

These limitations present opportunities for future research and the continued evolution of efficient exploration in reinforcement learning.

A Numerical Results

The appendix begins with the numerical results. We first introduce the Forked RiverSwim environment, a more complex variant of the traditional RiverSwim model.

Our discussion continues with a detailed exposition of Section 4.2, providing further experimental details. We conclude this section with additional findings related to both the tabular case and two specific problems: the CartPole Swing-Up and the Slipping DeepSea.

A.1 The Forked Riverswim Environment

The `Forked RiverSwim`(N) is a novel environment (see also Figure 7) where the agent needs to constantly explore two different states, (s_g, s'_g), to learn the optimal policy. The number of states is $2N-1$, and there are 3 actions.

The environment is similar to `RiverSwim`, but the initial state s_1 forks into two rivers: the final state in both branches of the rivers (s_g and s'_g) have a similar high reward. Furthermore, the agent can deterministically switch between the two branches at any intermediate state. Intermediate states do not give any reward. Moreover, a little subtlety is that the agent can exploit the deterministic transition between s_1 and s'_2 to deterministically transition to s_2 (although this has a small effect as N grows large).

Lastly, the Bernoulli rewards in s_g and s'_g, which are the *highly rewarding* states, are quite similar (1 vs 0.95). Therefore, an optimal policy that starts in s_1 should achieve a slightly better reward than the optimal policy on the `RiverSwim` environment with $N+1$ states (due to the fact that the transition to s_2 from s_1 can be made in a deterministic way).

Due to these reasons, this variant introduces additional complexity into the decision-making process. It is reasonable that a learning algorithm may learn an approximately good greedy policy in a short time-span, but not exactly the optimal one. In fact, we may expect an algorithm to take longer (compared to `Riverswim`) to learn the true optimal policy. Finally, always compared to `RiverSwim`, the sample complexity is of orders of magnitude higher, as also depicted in Figure 1. For a Python implementation, please refer to the GitHub repository of this manuscript.

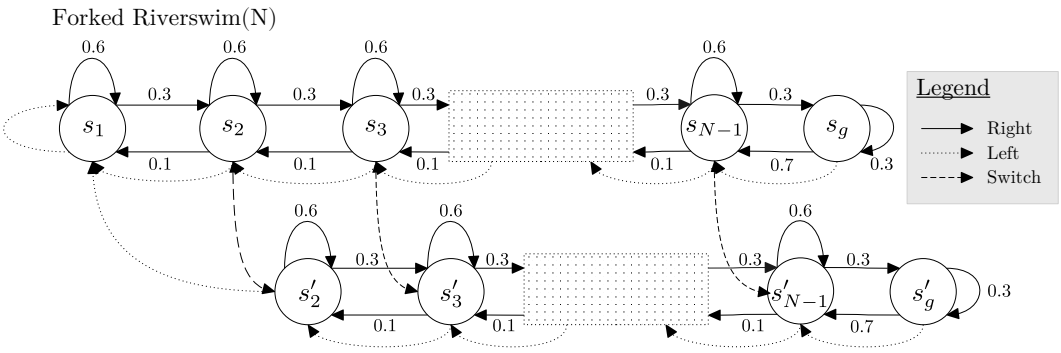

Figure 7: `Forked Riverswim`(N) with $|S| = 2N-1$ states. When taking action `left` in s_1 the agent observes a Bernoulli reward r of parameter 0.05. When taking action `right` in s_g (resp. s'_g) the agent observes a reward r drawn from a Bernoulli of parameter 1 (resp. 0.95). In all other states the reward is 0. Action `left` and `switch` are deterministic, while the probability of action `Right` is indicated in the figure. The square boxes indicate that the pattern of states is being repeated from s_3 (or s'_3) until s_{N-1} (or s'_{N-1}). This variant introduces additional complexity into the decision-making process, as the Bernoulli rewards in s_g and s'_g are quite similar (1 vs 0.95).

A.2 Details of Example 4.2

In the following we report the details of Section 4.2. In Section 4.2 we evaluated the characteristic time of three different environments with same discount factor $\gamma = 0.95$:

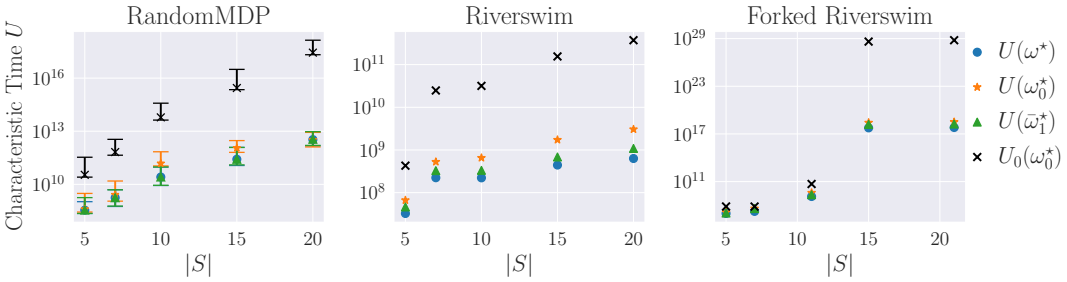

Figure 8: Comparison of (2) and (3) for discount $\gamma = 0.99$ and different sizes of the state space S. We evaluated different allocations using $U_0(\omega)$ and $U(\omega)$. The allocations are: $\omega^\star$ (the optimal allocation in Equation (3)), $\omega_0^\star$ (the optimal allocation in Equation (2)) and $\omega_1^\star$ (the optimal allocation that we get from (3) by setting $k = 1$ uniformly across states and actions). Results for the RandomMDP indicate the median and the bars 95% confidence intervals across 30 runs.

1. `RandomMDP`: an MDP with $|S|$ states and 3 actions. The transition probability for each (s, a) is drawn from a Dirichlet distribution $\mathrm{Dir}(\alpha_1, \ldots, \alpha_{|S|})$, with $\alpha_i = \alpha_{i-1} + (i-1)/10$ and $\alpha_1 = 1$. The rewards also follow the same Dirichlet distribution, that is, for each (s, a) we sample a $|S|$-dimensional vector q of rewards from $\mathrm{Dir}(\alpha_1, \ldots, \alpha_{|S|})$. This vector defines the rewards in the next state $r(s, a, s') = q_{s'}$, with $q \sim \mathrm{Dir}(\alpha_1, \ldots, \alpha_{|S|})$. For this type of environment see also the details of the instance-specific quantities in Table 1.

2. `RiverSwim`: this environment is specified in [60], but we refer to the version used in [38] for a direct comparison. The reward is always 0 except in the initial state s_1, and the final state $s_{|S|}$. In the initial state we have $q(1|s_1, \text{left}) = 0.05$ (probability 0.05 of observing a reward of 1, and 0.95 probability of observing a reward of 0), while in the final state $q(1|s_{|S|}, \text{right}) = 1$. All other rewards are set to 0. Transition probabilities are the same as in [60]. For this type of environment see also the details of the instance-specific quantities in Table 2.

3. `Forked RiverSwim`: we refer the reader to Appendix A.1 for a description of this environment. For this type of environment see also the details of the instance-specific quantities in Table 3.

Interestingly, these environments have different properties that make them suitable for analysis: (1) the `RandomMDP` environment has very small gaps and variances; (2) the `Riverswim` environment has a relatively larger maximum span; (3) the `Forked Riverswim` environment, in contrast to the `Riverswim` environment, has a very small minimum gap $\Delta_{\min}$ and similar values for the span.

$\lvert S\rvert$	$\Delta_{\min}$	$\max_{sa} \Delta_{sa}$	$\min_{sa} \mathrm{MD}_{sa}[V^\star]$	$\max_{sa} \mathrm{MD}_{sa}[V^\star]$	$\min_{sa} \mathrm{Var}_{sa}[V^\star]$	$\max_{sa} \mathrm{Var}_{sa}[V^\star]$	$\max_{s,a,k} M_{sa}^k[V^\star]^{2^{-k}}$
5	$1.1 \cdot 10^{-2}$	$1.6 \cdot 10^{-1}$	$6.4 \cdot 10^{-2}$	$1.0 \cdot 10^{-1}$	$8.3 \cdot 10^{-4}$	$3.4 \cdot 10^{-3}$	$1.0 \cdot 10^{-1}$
10	$2.3 \cdot 10^{-3}$	$6.3 \cdot 10^{-2}$	$2.7 \cdot 10^{-2}$	$3.6 \cdot 10^{-2}$	$1.4 \cdot 10^{-4}$	$3.7 \cdot 10^{-4}$	$3.6 \cdot 10^{-2}$
25	$1.2 \cdot 10^{-4}$	$1.0 \cdot 10^{-2}$	$4.6 \cdot 10^{-3}$	$5.1 \cdot 10^{-3}$	$3.3 \cdot 10^{-6}$	$4.9 \cdot 10^{-6}$	$5.1 \cdot 10^{-3}$
50	$9.5 \cdot 10^{-6}$	$1.9 \cdot 10^{-3}$	$9.1 \cdot 10^{-4}$	$9.5 \cdot 10^{-4}$	$1.2 \cdot 10^{-7}$	$1.4 \cdot 10^{-7}$	$9.5 \cdot 10^{-4}$
100	$1.1 \cdot 10^{-6}$	$3.7 \cdot 10^{-4}$	$1.8 \cdot 10^{-4}$	$1.8 \cdot 10^{-4}$	0	0	$1.8 \cdot 10^{-4}$

Table 1: Details of the instance-specific quantities for the `RandomMDP` environment (we evaluated up to $k = 19$). Results indicate an average over 300 different realizations. Confidence intervals are omitted for brevity, and values are rounded up to the 1st decimal.

$\lvert S\rvert$	$\Delta_{\min}$	$\max_{sa} \Delta_{sa}$	$\min_{sa} \mathrm{MD}_{sa}[V^\star]$	$\max_{sa} \mathrm{MD}_{sa}[V^\star]$	$\min_{sa} \mathrm{Var}_{sa}[V^\star]$	$\max_{sa} \mathrm{Var}_{sa}[V^\star]$	$\max_{s,a,k} M^k_{sa}[V^\star]^{2^{-k}}$
5	$7.6 \cdot 10^{-2}$	$1.3 \cdot 10^{0}$	$1.7 \cdot 10^{0}$	$3.0 \cdot 10^{0}$	0	$3.6 \cdot 10^{-1}$	$1.1 \cdot 10^{0}$
10	$3.4 \cdot 10^{-2}$	$1.3 \cdot 10^{0}$	$2.5 \cdot 10^{0}$	$4.5 \cdot 10^{0}$	0	$3.7 \cdot 10^{-1}$	$1.1 \cdot 10^{0}$
25	$1.9 \cdot 10^{-2}$	$1.3 \cdot 10^{0}$	$2.5 \cdot 10^{0}$	$5.0 \cdot 10^{0}$	0	$3.7 \cdot 10^{-1}$	$1.1 \cdot 10^{0}$
50	$8.4 \cdot 10^{-3}$	$1.3 \cdot 10^{0}$	$2.7 \cdot 10^{0}$	$5.4 \cdot 10^{0}$	0	$3.7 \cdot 10^{-1}$	$1.1 \cdot 10^{0}$
100	$2.1 \cdot 10^{-4}$	$1.3 \cdot 10^{0}$	$2.9 \cdot 10^{0}$	$5.5 \cdot 10^{0}$	0	$3.7 \cdot 10^{-1}$	$1.1 \cdot 10^{0}$

Table 2: Details of the instance-specific quantities for the `Riverswim` environment (we evaluated up to $k = 19$). Values are rounded up to the 1st decimal.

$\lvert S\rvert$	$\Delta_{\min}$	$\max_{sa} \Delta_{sa}$	$\min_{sa} \mathrm{MD}_{sa}[V^\star]$	$\max_{sa} \mathrm{MD}_{sa}[V^\star]$	$\min_{sa} \mathrm{Var}_{sa}[V^\star]$	$\max_{sa} \mathrm{Var}_{sa}[V^\star]$	$\max_{s,a,k} M^k_{sa}[V^\star]^{2^{-k}}$
5	$1.0 \cdot 10^{-1}$	$1.4 \cdot 10^{0}$	$1.0 \cdot 10^{0}$	$2.0 \cdot 10^{0}$	0	$3.2 \cdot 10^{-1}$	$1.0 \cdot 10^{0}$
11	$2.8 \cdot 10^{-2}$	$1.3 \cdot 10^{0}$	$1.6 \cdot 10^{0}$	$2.9 \cdot 10^{0}$	0	$4.9 \cdot 10^{-1}$	$2.0 \cdot 10^{0}$
25	$1.0 \cdot 10^{-6}$	$1.3 \cdot 10^{0}$	$1.7 \cdot 10^{0}$	$3.2 \cdot 10^{0}$	0	$4.8 \cdot 10^{-1}$	$2.0 \cdot 10^{0}$
51	$1.0 \cdot 10^{-6}$	$1.3 \cdot 10^{0}$	$2.1 \cdot 10^{0}$	$4.2 \cdot 10^{0}$	0	$4.8 \cdot 10^{-1}$	$2.0 \cdot 10^{0}$
101	$1.0 \cdot 10^{-6}$	$1.3 \cdot 10^{0}$	$2.5 \cdot 10^{0}$	$5.1 \cdot 10^{0}$	0	$4.8 \cdot 10^{-1}$	$2.0 \cdot 10^{0}$

Table 3: Details of the instance-specific quantities for the `Forked Riverswim` environment (we evaluated up to $k = 19$). Values are rounded up to the 1st decimal.

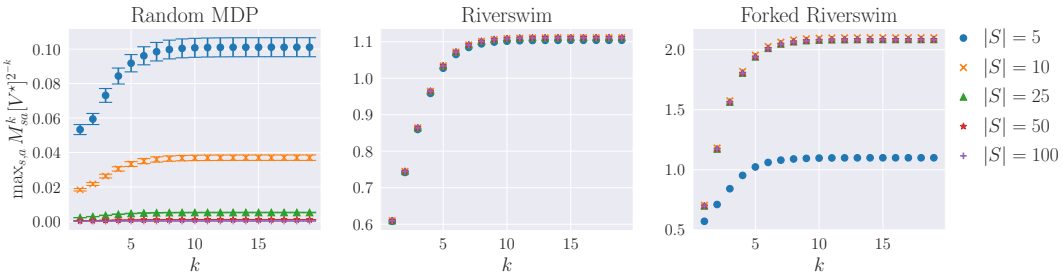

Figure 9: Plot of $\max_{s,a} M^k_{sa}[V^\star]^{2^{-k}}$ for various values of k. For the random MDP we depict the median value, as well as the 95% confidence interval.

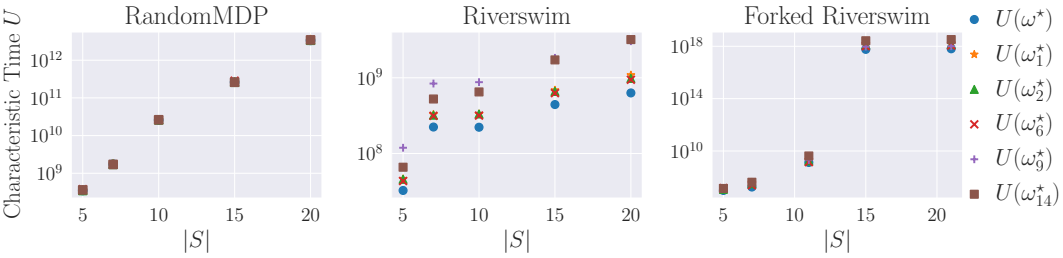

Figure 10: Evaluation of $\omega^\star_k$ for different values of k. For the `RandomMDP` we only show the median value over 300 runs.

Finally, in Figure 9, we depict $\max_{s,a} M^k_{sa}[V^\star]^{2^{-k}}$ for different values of k, up to $k = 19$. For the `RandomMDP` environment we observe that $\max_{s,a} M^k_{sa}[V^\star]^{2^{-k}}$ tends to the maximum of the span $\max_{sa} \mathrm{MD}_{sa}[V^\star]$, which depends on the size of the state space (as $|S|$ grows larger the span diminishes). For the other two environments, `Riverswim` and `Forked Riverswim`, $\max_{s,a} M^k_{sa}[V^\star]^{2^{-k}}$ does not seem to depend on the size of the state space. Furthermore, we also observe a sudden convergence of this quantity for relatively small values of k, followed by a relatively very slow increase.

In Figure 10 are shown the results when we evaluate the allocations $\omega^\star_k$ for different values of k. In general, we do not observe a striking difference between those allocations.

A.3 Riverswim and Forked Riverswim - Description and Additional Results

In Figure 11 we present results from the Riverswim and ForkedRiverswim environments. These results include data from two new algorithms: O-BPI (Online Best Policy Identification) and PS-MDP-NAS (Posterior Sampling for MDP-NaS).

O-BPI is a novel algorithm that draws inspiration from MDP-NAS. However, a distinguishing characteristic is its use of stochastic approximation to determine the Q-values and M-values. These values, as for MF-BPI, are used to compute the allocation ω by solving the sample-complexity bound $\inf_{\omega \in \Omega(\phi)} U(\omega)$ with navigation constraints. On the other hand, PS-MDP-NAS is an adaptation of MDP-NAS that uses posterior sampling over the MDP's model to address the parametric uncertainty. It's worth noting that both these algorithms, O-BPI and PS-MDP-NAS, are model-based, and a detailed description of these algorithms is available in the following section, see **??** and Algorithm 3.

The results in Figure 11 clearly show the superiority of these allocation computing methods compared to other algorithms such as PSRL and Q-UCB.

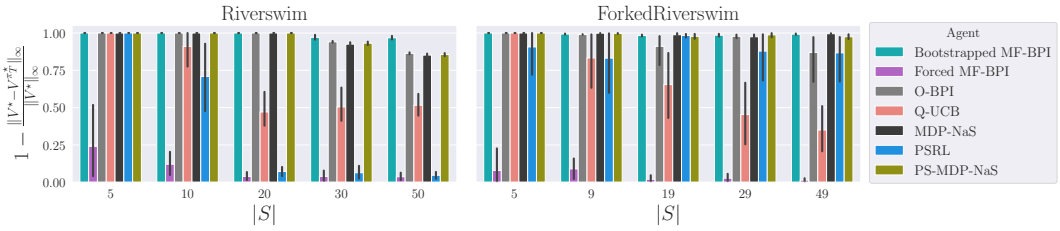

Figure 11: Evaluation of the estimated optimal policy $\pi_T^\star$ after T steps for MF-BPI, O-BPI, Q-UCB, MDP-NAS, PS-MDP-NAS, and PSRL. Results are averaged across 10 seeds and lines indicate 95% confidence intervals. Note that for `Forked Riverswim` we have $N = 2|S| - 1$.

Figure 12 on the next page provides a visualization of the performance of each algorithm over the entire horizon $t = 0, \ldots, T-1$. We exhibit the performance of the estimated greedy policy $\pi_t^\star$ at each timestep t for each respective method. The results offer a clear demonstration of the efficiency of those methods based on the instance-specific sample complexity lower bound.

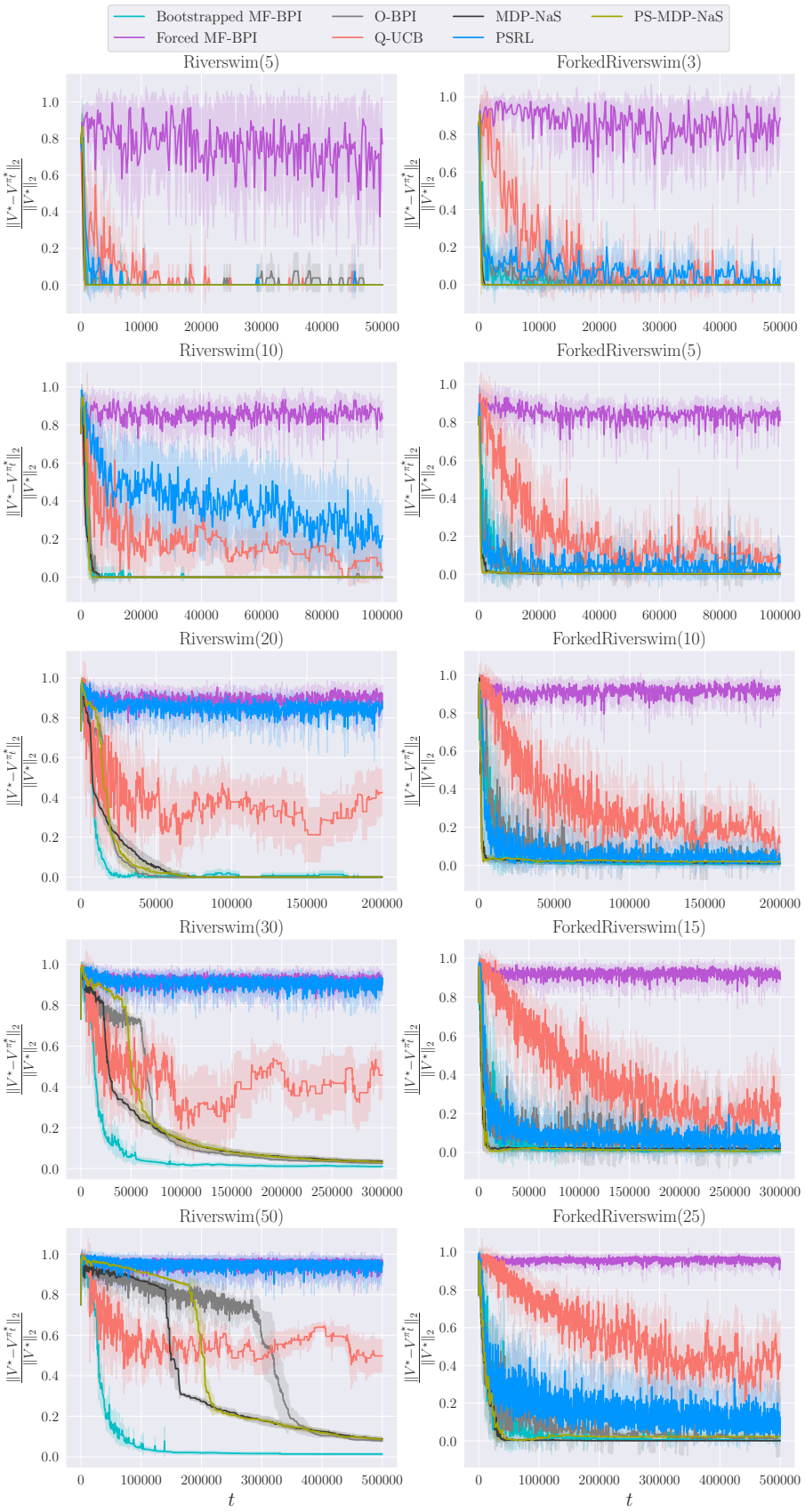

Figure 12: Evaluation of the estimated optimal policy $\pi_t^\star$ for MF-BPI, O-BPI, Q-UCB, MDP-NaS, PS-MDP-NaS, and PSRL. Results are averaged across 10 seeds and lines indicate 95% confidence intervals.

A.4 Slipping DeepSea - Description and Additional Results

Description. The Slipping DeepSea problem is an hard-exploration reinforcement learning problem. In the standard version, there's an $N \times N$ grid, and the agent starts in the top left corner (state 0, 0) and needs to reach the bottom right corner (state $N-1$, $N-1$) for a large reward (the state vector is an N^2-dimensional vector, that one-hot encodes the agent's position in the grid). The agent can move diagonally, left or right (or down when close to the wall). The agent incurs in a cost when moving of $0.01/N$, while obtaining a positive reward of 1 when reaching the bottom right corner. Furthermore, we introduce the modification that there is a small probability of 0.05 that the incorrect action will be executed. This is a challenging problem because the optimal policy requires the agent to move (incurring a negative reward) many times before eventually reaching the high reward in the bottom right corner. However, due to the stochastic nature of the problem (the chance of slipping), the agent might be forced to take suboptimal actions, making it harder to learn the optimal policy.

Additional results. Figure 13 presents additional metrics encapsulating the exploration conducted by each algorithm, offering a comprehensive summary of the exploration process after T episodes for each size N (note that for a given size N the number of input features in the state is N^2).

We focus on two key metrics: (a) $(t_{\text{visit}})_{ij}$ and (b) $(t_{\text{avg}})_{ij}$. Here, (a) $(t_{\text{visit}})_{ij}$ represents the last timestep t at which a cell (i, j) was visited (this value is normalized by NT, the multiplication of the grid size and the number of episodes), while (b) $(t\text{avg})_{ij}$ signifies the average frequency with which a cell (i, j) was visited. In terms of arrangement, from the top downwards: (1) we present

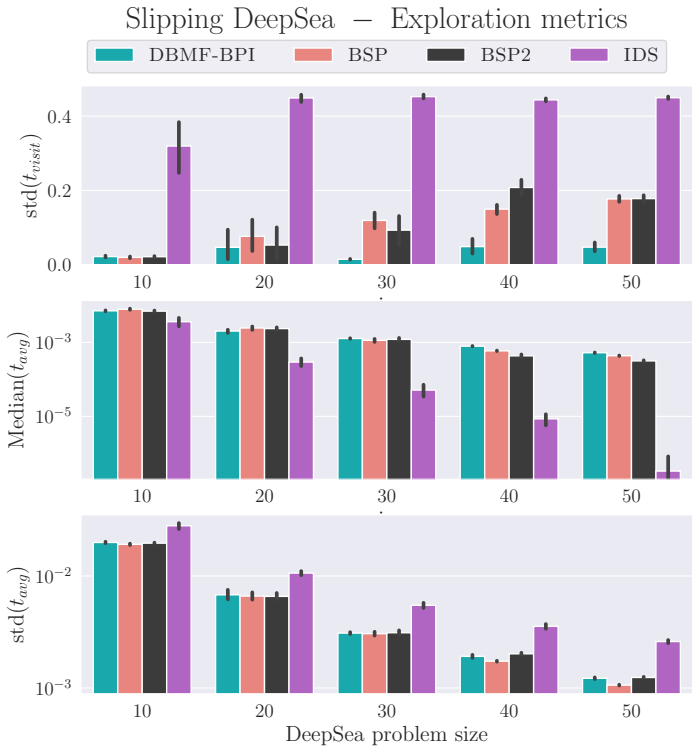

Figure 13: Slipping DeepSea problem - exploration metrics. From top to bottom: (1)standard deviation of t_{visit} at the last episode, depicting how much each agent explored (the lower the better); (2) median value of $(t_{\text{avg}})_{ij}$, *i.e.*, the median value of a cell's visit frequency; (3) standard deviation of $(t_{\text{avg}})_{ij}$ across all cells. Results are averaged over 24 runs and bars indicate 95% confidence intervals.

the standard deviation of $(t_{\text{visit}})_{ij}$ across all cells; (2) we show the median value of a cell's visit frequency; (3) we depict the standard deviation of $(t_{\text{avg}})_{ij}$ across all cells. From the central plot, we notice that DBMF-BPI tends to visit all cells slightly more frequently. The first plot also highlights that DBMF-BPI maintains a consistent visit rate to all cells. This pattern is a strong indication of DBMF-BPI's explorative behavior. Conversely, neither BSP nor BSP2 match this performance in

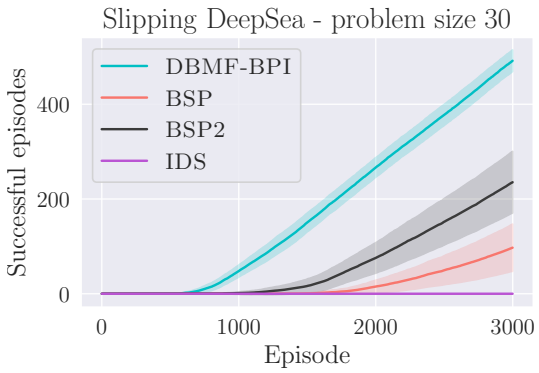

Figure 14: Slipping DeepSea problem. Total number of successful episodes (*i.e.*, that the agent managed to reach the final reward) for a grid with 30^2 input features.

terms of successful episodes (shown in Figure 14), despite the median value of t_{avg} being very similar to that of DBMF-BPI. In order to provide a more comprehensive view, Figure 15 and Figure 16 present additional exploration metrics. Specifically, we display $(t_{\text{avg}})_{ij}$ and $(t_{\text{visit}})_{ij}$, respectively, after $T = 3000$ episodes, given a DeepSea problem size of 30. The initial plot illustrates how DBMF-BPI tends to concentrate on the grid's diagonal. However, the bottom plot shows that, in spite of this diagonal focus, DBMF-BPI also maintains a consistent exploration of other cells within the grid. We also observe how BSP seems to uniformly explore all cells, while IDS does not manage to explore the entire grid within the number of episodes. Last, but not least, on the right column in

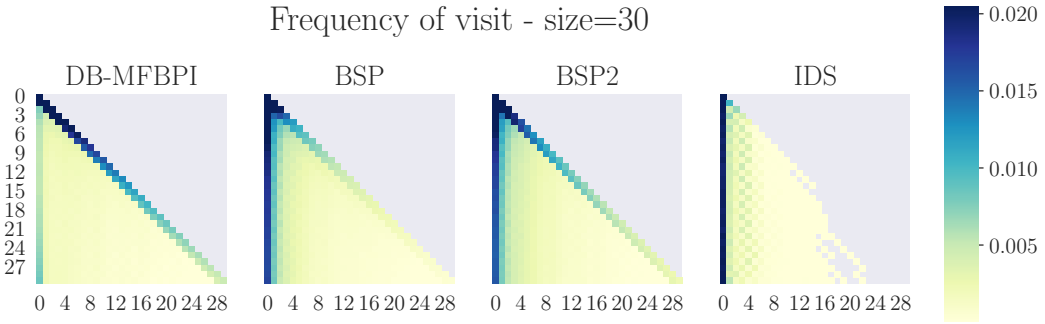

Figure 15: Slipping DeepSea problem. In this figure we depict the average frequency of visits, after 3000 episodes, when the size of the problem is $k = 30$.

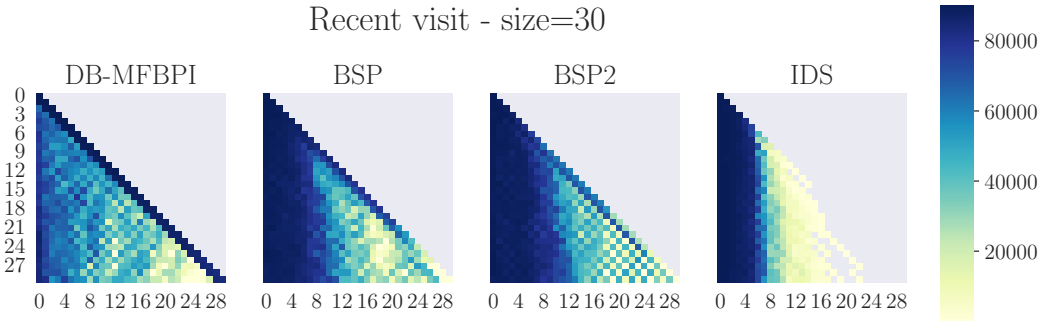

Figure 16: Slipping DeepSea problem. In this figure we depict the last timestep a cell was visited, after 3000 episodes, when the size of the problem is $k = 30$.

Figure 17, are shown the results for the learnt greedy policy $\pi_t^\star$ at time t. Clearly, DBMF-BPI is able to learn an efficient policy more quickly than the other methods for different problem sizes.

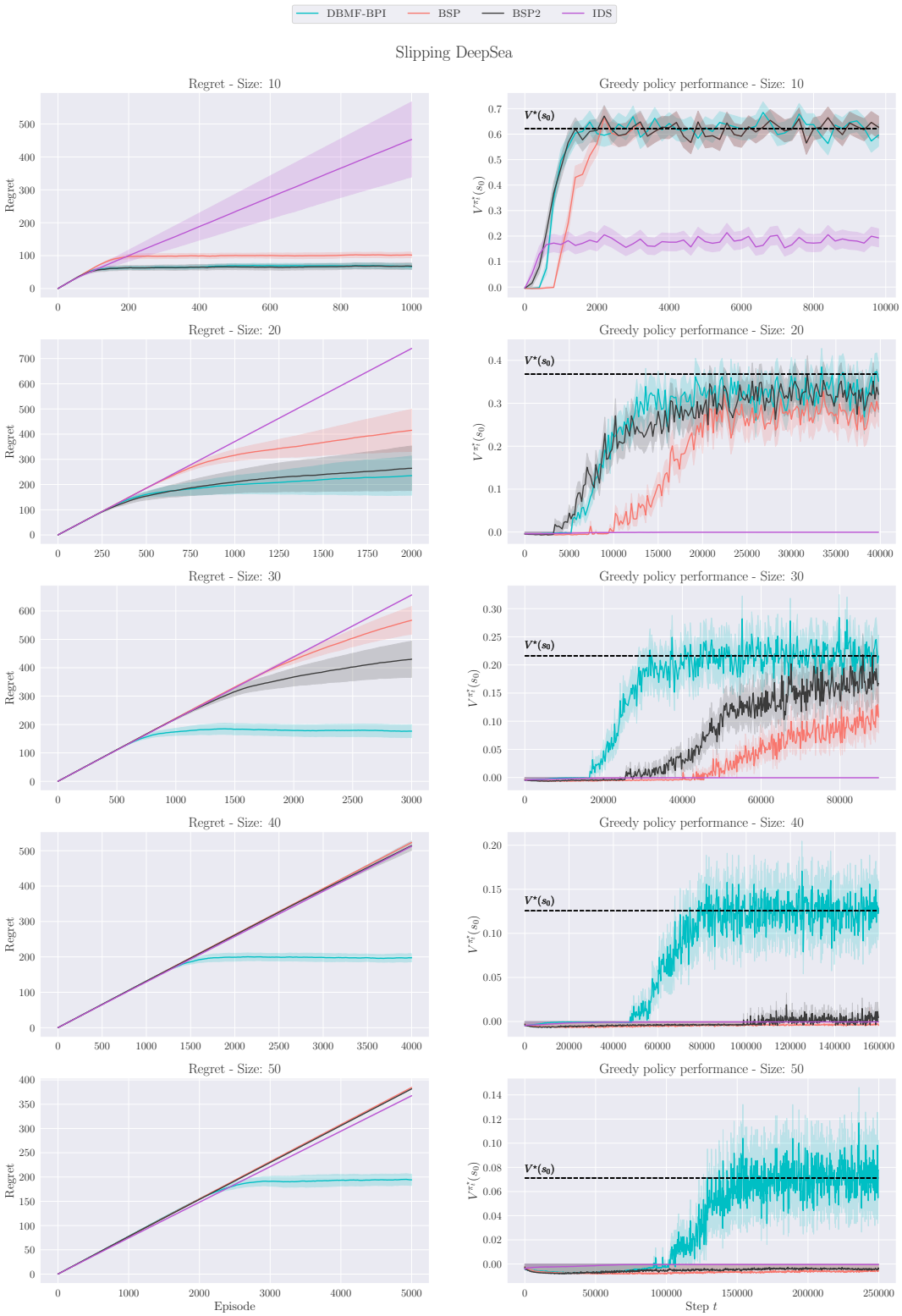

Figure 17: Slipping DeepSea problem - evaluation of the greedy policy. On the left we depict the regret of the learning agent over the number of episodes T for each problem size k. On the right, we display the average value of the learnt greedy policy $\pi_t^\star$ at time t (black dashed-line indicates the average optimal value). Results are averaged over 24 runs, and the shaded area depicts 95% confidence intervals.

A.5 Cartpole Swingup - Description and Additional Results

Description. In this subsection we present additional results for the Cartpole swingup problem. The cartpole swingup problem is a classic problem in control theory and reinforcement learning [9]. The task is to balance a pole that is attached by an un-actuated joint to a cart, which moves along a frictionless track. The system is controlled by applying a force to the cart. Initially, the pole is hanging down and the goal is to swing it up so it stays upright. In contrast to the classic cartpole balance problem, the pole needs not only to be balanced when it's upright but also to be swung up to the upright position.

The state of the system at any point in time is described by four variables: the position of the cart x, the velocity of the cart $\dot{x}$, the angle of the pole θ, and the angular velocity of the pole $\dot{\theta}$. There are 4 additional variables in the state, and for simplicity we refer the user to [52].

To make the problem more difficult, as in [52] we introduce a parameter $k \in \{1, \ldots, 19\}$ (to not be confused with the parameter of $M^k_{sa}[V^\star]$) that parameterizes the reward function. Specifically, the agent observes a positive reward of 1 only if the pole's angle satisfies $\cos(\theta) > k/20$, and the cart's position satisfies $|x| \leq 1 - k/20$. There is also a negative reward of -0.1 that the agent incurs for moving, which aggravates the explore-exploit tradeoff (algorithms like DQN [41] simply remain still).

Additional results. In Figures 18 to 20, we provide supplementary results for this problem. Figure 18 illustrates the total upright time achieved by each learner after 200 episodes, across various difficulty levels, k. Here, the total upright time refers to the total count of steps where the pole maintained an angle satisfying $\cos(\theta) > k/20$, concurrently with the cart maintaining a position that satisfied $|x| \leq 1 - k/20$.

Subsequently, Figure 19 showcases the evolution of this metric throughout all 200 episodes.

Figure 20 demonstrates the performance of the learnt greedy policy $\pi^\star_t$ over the course of the training. Every 10 episodes, we evaluated the greedy policy over 20 episodes and computed the cumulative reward.

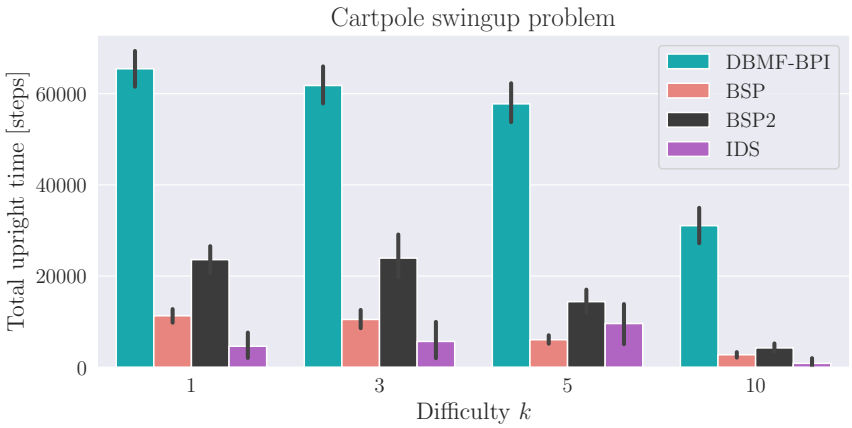

Figure 18: Cartpole swingup problem. Total upright time after 200 episodes for different difficulties k. To observe a positive reward, the pole's angle must satisfy $\cos(\theta) > k/20$, and the cart's position should satisfy $|x| \leq 1 - k/20$. Bars indicate 95% confidence intervals.

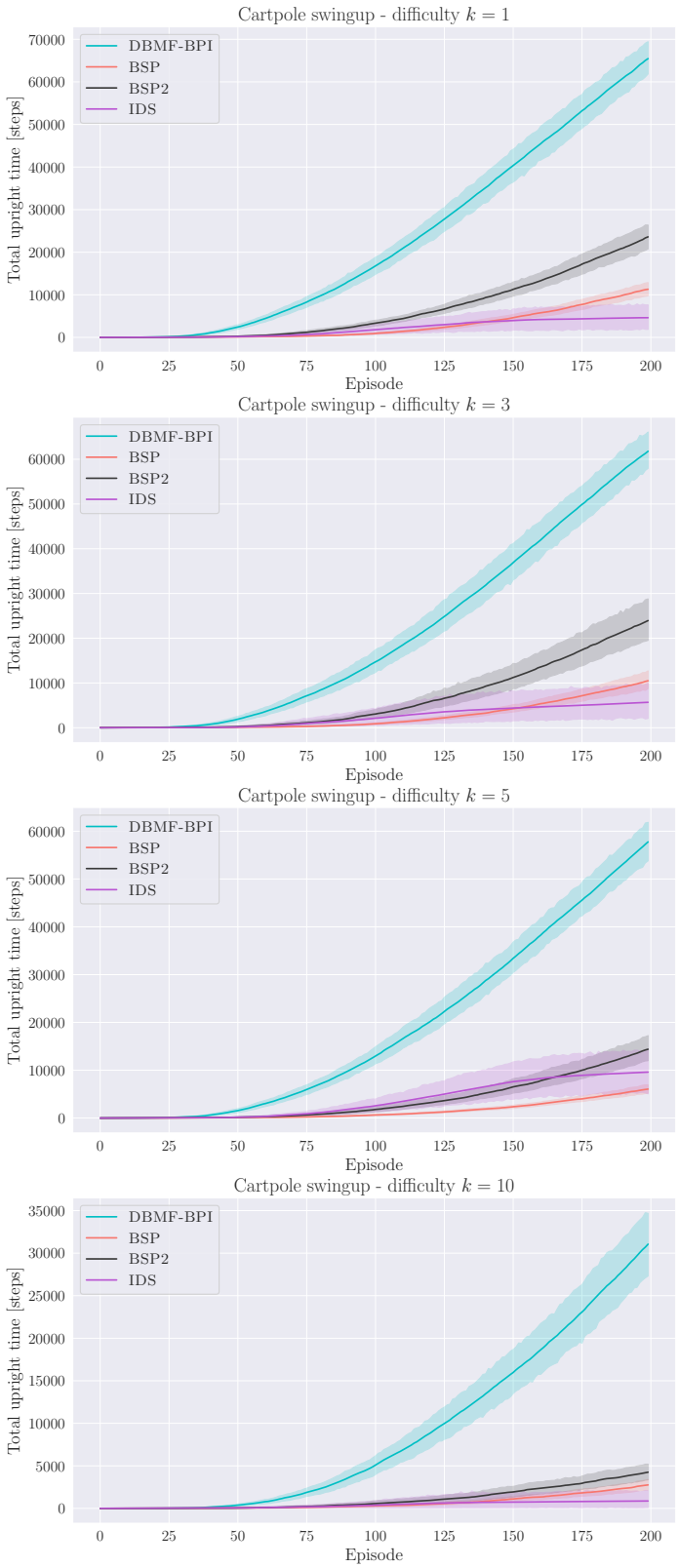

Figure 19: Cartpole swingup problem. Total upright time over 200 episodes for different difficulties k. To observe a positive reward, the pole's angle must satisfy $\cos(\theta) > k/20$, and the cart's position should satisfy $|x| \leq 1 - k/20$. Bars indicate 95% confidence intervals.

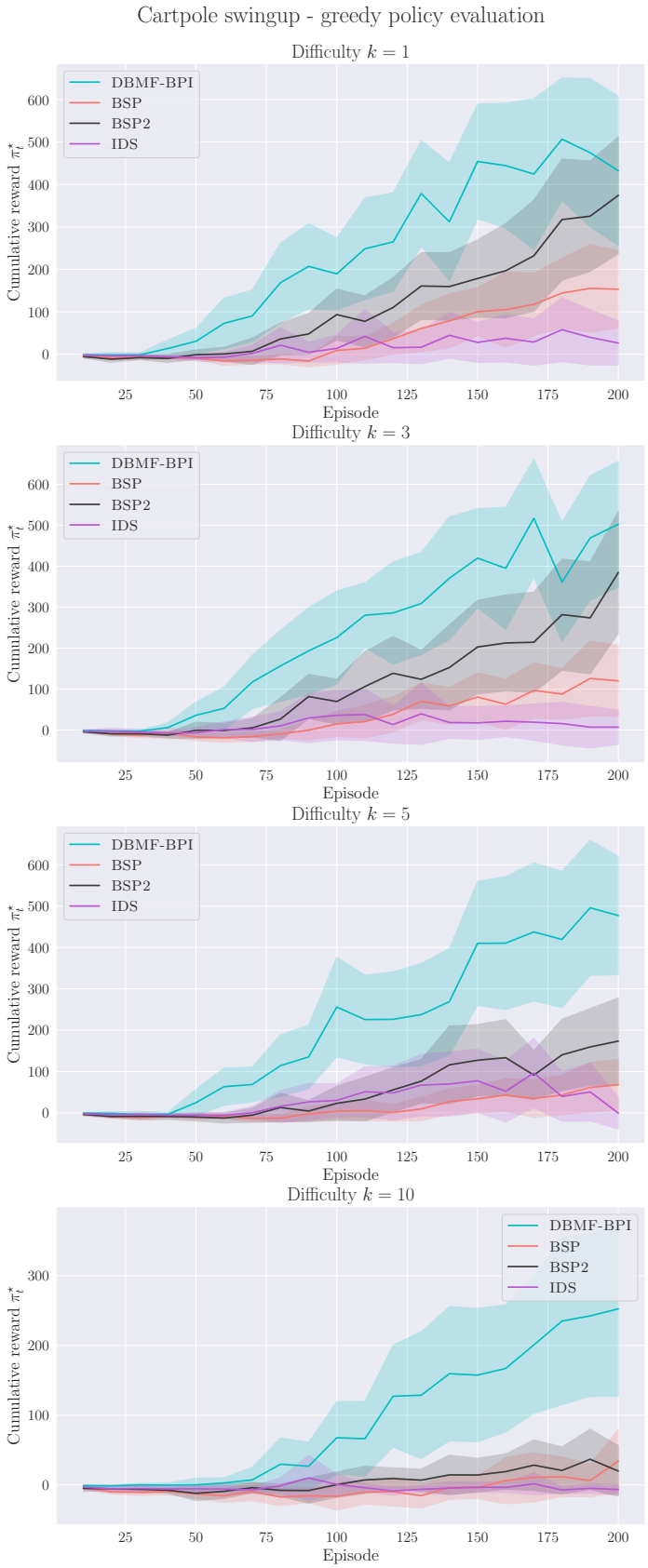

Figure 20: Cartpole swingup problem. Performance of the learnt greedy policy $\pi_t^\star$ over the training episodes (average cumulative reward collected by the greedy policy).

Exploration results. In Figures 21 to 23, we show additional results that illustrate the exploration of the various algorithms for difficulties $k = 3, 5$.

In Figure 21, we display two metrics at each training step t: the entropy of visit frequency and the entropy of the most recent visit. The first metric quantifies how thoroughly the method has explored the state space $(x, \dot{x}, \theta, \dot{\theta})$ up to time t. To do this, we discretize the state space into bins and tally the occurrences in each bin. We then normalize these counts by their sum and calculate the resulting entropy, which is normalized to the range $[0, 1]$.

While this measure of visit frequency provides some insight, it is insufficient for understanding whether the algorithm continues to explore new states or revisits old ones. To address this, the second metric measures the dispersion of the timing of the last visits to various regions of the state space. A larger dispersion indicates that the algorithm is concentrating on a specific region, resulting in a smaller entropy (and vice-versa). To calculate this, we again use normalized entropy.

Finally, in Figure 22 and Figure 23, we illustrate the visitation frequency after $20K$ training steps for $(x, \dot{x})$ and $(\dot{x}, \dot{\theta})$ at difficulty levels $k = 3$ and $k = 5$. Darker regions signify higher visitation frequencies. The pattern in $(\dot{x}, \dot{\theta})$ is characteristic of algorithms that have learned to stabilize the policy. Notably, DBMF-BPI is also actively exploring various velocities. A similar trend is observed for $(x, \dot{x})$: while most methods focus on an s-shaped trajectory, DBMF-BPI also explores other regions of the state space.

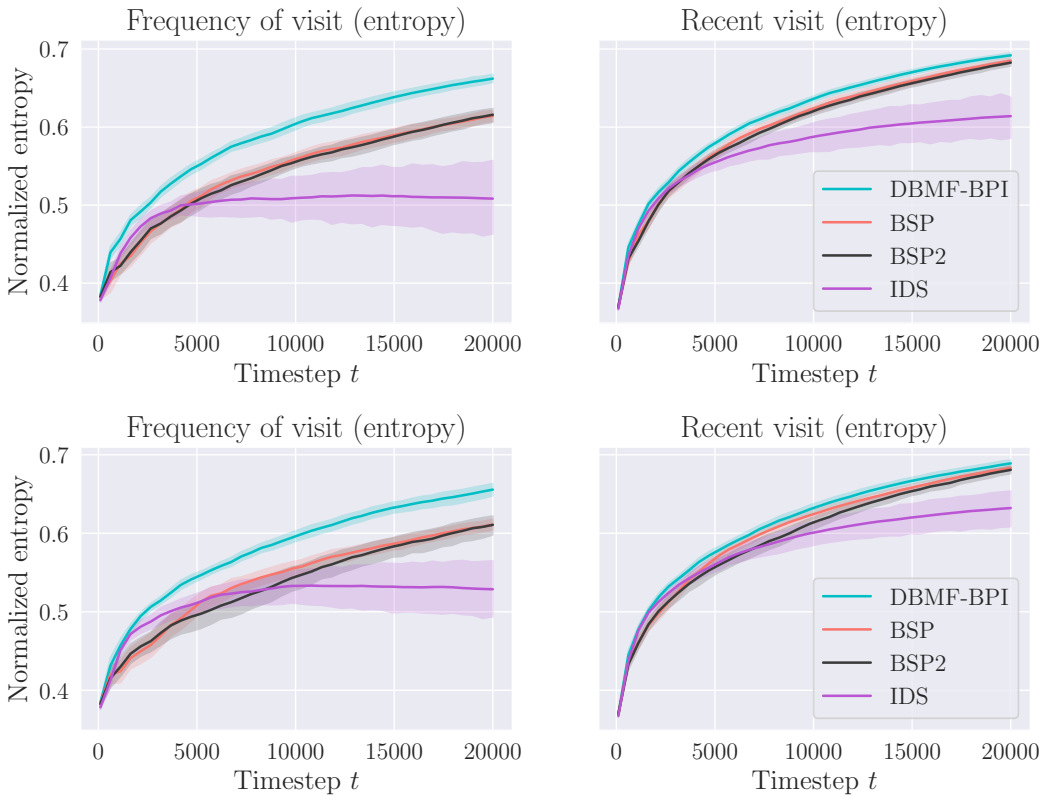

Figure 21: Exploration in Cartpole swingup: At the top, we present results for difficulty $k = 3$, and at the bottom, for $k = 5$. In the left column, we depict the entropy of visitation frequency for the state space $(x, \dot{x}, \theta, \dot{\theta})$ during training. In the right column, we display a measure of the dispersion of the most recent visits; smaller values indicate that the agent is less explorative as t increases.

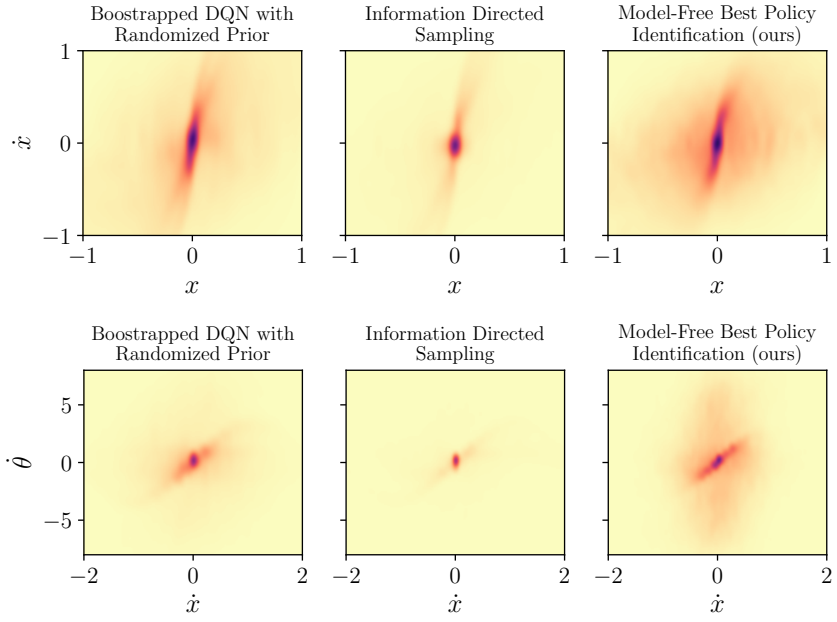

Figure 22: Cartpole Swingup [52] after 20K training steps for difficulty $k = 3$, comparing BSP (Bootstrapped DQN with randomized priors) [50], IDS (Information-Directed Sampling) [44], and MF-BPI (Model-Free Best Policy Identification). Darker areas indicate higher visitation frequency. At the top we show this frequency for $(x, \dot{x})$, the cart's position and linear's velocity, and at the bottom of $(\dot{x}, \dot{\theta})$, the cart's linear and pole's angular velocities.

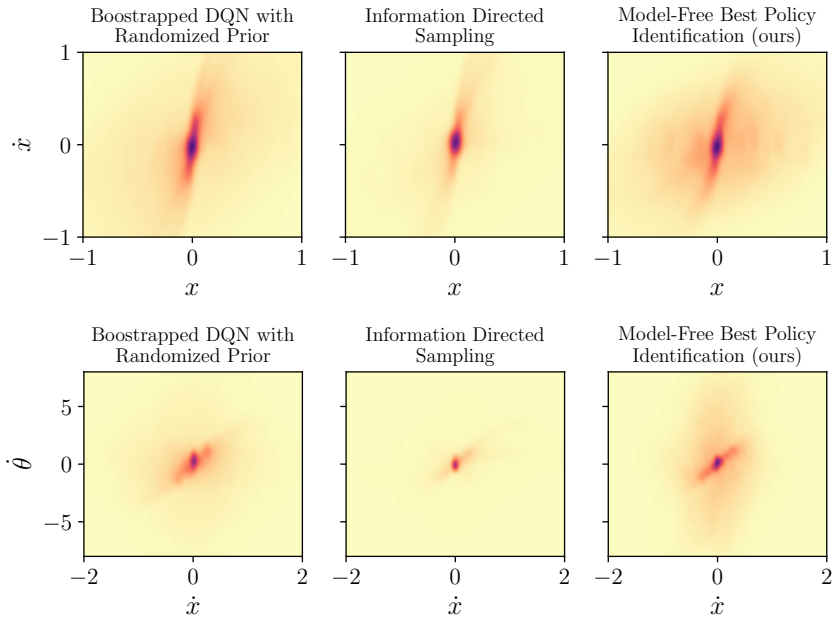

Figure 23: Cartpole Swingup [52] after 20K training steps for difficulty $k = 5$, comparing BSP (Bootstrapped DQN with randomized priors) [50], IDS (Information-Directed Sampling) [44], and MF-BPI (Model-Free Best Policy Identification). Darker areas indicate higher visitation frequency. At the top we show this frequency for $(x, \dot{x})$, the cart's position and linear's velocity, and at the bottom of $(\dot{x}, \dot{\theta})$, the cart's linear and pole's angular velocities.

A.6 Parameters, Hardware, Code and Libraries

In this section, we outline the parameters used for the simulations, describe the hardware employed to run the simulations, and list the libraries that we used.

A.6.1 Simulation parameters - Riverswim and Forked Riverswim

In both the Riverswim and Forked Riverswim environments we used a discount factor of $\gamma = 0.99$. Depending on the size of the state space, the horizon length was different. We used $T = 10000 \times |S|$ for the Riverswim environment, and $T = 20000 \times N$ for the Forked Riverswim environment (where N is the length of the main river; see also the description of the environment in Appendix A.1).

We run simulations for 10 different seeds, and evaluated the estimated greedy policy $\pi_t^\star$ every 200 steps. All agents were optimistically initialized (*i.e.*, the Q-values were initialized to $1/(1-\gamma)$, etc...), and model-based approaches used additive smoothing (with factor 1).

For the MDP-NAS and PS-MDP-NAS (see next section for a description) we computed the allocation every $T_0 = \min(T_{max}, \max(200, \frac{T_{max} t}{T/2}))$ steps, where $T_{max} = \frac{2000T}{50000}$. For PSRL we computed a new greedy policy every $\lceil 1/(1-\gamma) \rceil$ steps.

We used a learning rate of $\alpha_t = \frac{H+1}{H+k_t}$ to learn the Q-values, where $H = (1-\gamma)^{-1}$ and $k_t = N_t(s_t, a_t)$ is the number of visits to (s_t, a_t) at time t. Similarly, to learn the M-values we used a learning rate of $\beta_t = \alpha_t^{1.1}$ (which was not optimized).

For bootstrapped MFBPI we used a parameter $k = 1$, and an ensemble size $B = 50$ with training probability $p = 0.7$. Similarly, these values were not optimized.

All methods that employed an ϵ-soft policy, obtained a final policy ω by mixing mixed the original policy π with a uniform policy as follows $\omega(a|s) = (1-\epsilon_t)\pi(a|s) + \epsilon_t/|A|$. The value of ϵ_t is $\epsilon_t = \min(1, 1/N_t(s_t))$ where $N_t(s_t)$ is the total number of visits to state s_t at time t.

A.6.2 Simulation parameters - Slipping DeepSea

For the DeepSea problem we used a discount factor of $\gamma = 0.99$, and different problem sizes $N \in \{10, 20, 30, 40, 50\}$. The number of training episodes was $T = 100N$. Every 200 steps we evaluated the performance of the estimated greedy policy $\pi_t^\star$ over 20 episodes. For all simulations we used a slipping probability of 0.05. The number of features in the state is N^2, and the number of actions is 2.

Refer to Table 4 for the parameters of the agents.

Table 4: Parameters of the agents for the slipping DeepSea problem.

Property	DBMF-BPI	BSP	BSP 2	IDS
Ensemble size Q	20	20	20	$20 + \frac{(N-10)}{2}$
Ensemble size M	20	N.A.	N.A.	N.A.
Hidden layers sizes	[32]	[32]	[32]	[50]
Num. of quantiles	N.A.	N.A.	N.A.	50
Prior scale Q-values (depends on N)	$\{3, 5, 10, 15, 20\}$	$\{3, 5, 10, 15, 20\}$	$\{3, 5, 10, 15, 20\}$	N.A.
Prior scale M-values (depends on N)	$\{3, 5, 10, 15, 20\}$	$\{3, 5, 10, 15, 20\}$	$\{3, 5, 10, 15, 20\}$	N.A.
Replay buffer size	10^5	10^5	10^5	10^5
Training period	1	1	1	1
Target network update period	4	4	4	4
Batch size	128	128	128	128
Mask probability p	0.7	0.5	0.7	N.A.
Learning rate Q-values	5×10^{-4}	10^{-3}	10^{-3}	5×10^{-4}
Learning rate M-values	5×10^{-4}	N.A.	N.A.	N.A.
Learning rate quantile network	N.A.	N.A.	N.A.	10^{-6}
k	2	N.A.	N.A.	N.A.

A.6.3 Simulation parameters - Cartpole Swingup

For the Cartpole swingup problem we used a discount factor of $\gamma = 0.99$, and different difficulties $k \in \{1, 3, 5, 10\}$. The number of training episodes was $T = 200$, and we run simulations for 20 different seeds. Every 10 steps in the training we evaluated the performance of the estimated greedy policy $\pi_t^\star$ over 20 episodes. The state is a vector in $\mathbb{R}^8$ and the number of actions is 3.

For every method, with the exception of IDS, we set up the parameters in the i^{th} layer of each network by sampling from a truncated Gaussian distribution with a 0 mean and a standard deviation of $1/\sqrt{f_{in}}$, where f_{in} represents the number of inputs to the i^{th} layer. Values were cut off at twice the standard deviation. For IDS, enhancing the standard deviation improved results. Specifically, we employed a standard deviation of $1.5/\sqrt{f_{in}}$ for the Q-networks ensemble, and a standard deviation of $2/\sqrt{f_{in}}$ for the quantile network. Generally, this initialization mirrors an optimistic initialization. However, the results can vary significantly between runs, and our observation was that the IDS method often exhibited greater variance compared to the other methods incorporated in our study. To conclude, the bias for all layers was set to 0.

Refer to Table 5 for the parameters of the agents.

Table 5: Parameters of the agents for the Cartpole swingup problem.

Property	DBMF-BPI	BSP	BSP 2	IDS
Ensemble size Q	20	20	20	20
Ensemble size M	20	N.A.	N.A.	N.A.
Hidden layers sizes	[50]	[50]	[50]	[50]
Num. of quantiles	N.A.	N.A.	N.A.	50
Prior scale Q-values (depends on N)	3	3	3	N.A.
Prior scale M-values (depends on N)	3	3	3	N.A.
Replay buffer size	10^5	10^5	10^5	10^5
Training period	1	1	1	1
Target network update period	4	4	4	4
Batch size	128	128	128	128
Mask probability p	0.7	0.5	0.7	N.A.
Learning rate Q-values	5×10^{-4}	5×10^{-4}	5×10^{-4}	5×10^{-4}
Learning rate M-values	5×10^{-4}	N.A.	N.A.	N.A.
Learning rate quantile network	N.A.	N.A.	N.A.	10^{-6}
k	2	N.A.	N.A.	N.A.

A.6.4 Hardware and simulation time

To run the simulations, we used a local stationary computer with Ubuntu 20.10, an Intel® Xeon® Silver 4110 Processor (8 cores) and 48GB of ram. On average, it takes approximately 14 days to complete all the simulations contained in this manuscript. Ubuntu is an open-source Operating System using the Linux kernel and based on Debian. For more information, please check `https://ubuntu.com/`.

A.6.5 Code and libraries

We set up our experiments using Python 3.10 [67] (For more information, please refer to the following link `http://www.python.org`), and made use of the following libraries: Cython [10], NumPy [25], SciPy [68], PyTorch [53], CVXPY [20], MOSEK [1], Seaborn [72], Pandas [40], Matplotlib [27]. In the code, we make use of some code from the Behavior suite [52], which is licensed with the APACHE 2.0 license. Changes, and new code, are published under the MIT license. To run the code, please, read the attached README file for instructions.

B Algorithms

In the following section we describe the algorithms that we discuss in this manuscript. For simplicity, we provide a brief summary of them in form of table I.

Table 6: Description of the various algorithms

Name	Description	Key points
PS-MDP-NAS	An adaptation of MDP-NAS that uses posterior sampling to sample an MDP ϕ_t, which is then used to compute the optimal allocation (in Equation (3)).	Requires the user to: • Keep an estimate of the MDP. • Perform value/policy iteration. • Compute the allocation (a convex problem). • Uses posterior sampling at each time step to sample an MDP and compute the allocation.
O-BPI	An adaptation of MDP-NAS that learns the Q-values and M-values. These values are used to compute the optimal allocation in Equation (3).	• Does not perform value iteration. • Requires to keep an estimate of the transition function. • Compute the allocation (a convex problem). • Uses forced exploration to sample all state-action pairs i.o.
Bootstrapped MF-BPI	This algorithm is an extension of O-BPI that computes the allocation using the closed form solution in Proposition 5.1. The Q, M-values used to compute the allocation are bootstrap samples.	• Does not perform value iteration and does not require to keep an estimate of the model. • Closed form solution for the allocation. • Uses bootstrapping (forced exploration not necessary).
DBMF-BPI	An extension of BO-MFPI to the Deep-RL setting. The baseline architecture is inspired from BootstrappedDQN with prior networks. This architecture is then adapted to compute a generative allocation.	• Like boostrapped MF-BPI. • Requires to keep an ensemble of Q, M-networks. • Can be applied to continuous state spaces.

B.1 PS-MDP-NAS - Posterior Sampling for navigating MDPs

In this sub-section we present PS-MDP-NAS, an adaptation of MDP-NAS that uses posterior sampling. An outline of the algorithm is given in Algorithm 3. For simplicity, we omit the use of any stopping rule, since we focus more on the practical implementation of the algorithm.

At each timestep we sample an MDP ϕ_t from a posterior distribution, and use it to solve the optimal allocation in Theorem 4.2 with navigation constraints. When computing the optimal allocation $\arg\inf_{\omega \in \Omega(\phi)} U(\omega)$, we limit the maximum number of possible values of k for simplicity.

The algorithm considers a Dirichlet prior for the transition function, and a Gamma prior for the reward distribution. Specifically, for each (s, a) we have a prior hyper-parameter $\rho_{sa} \in \mathbb{R}^{|S|}$ that characterizes the transition function, and two other hyper-parameters $\alpha_{sa}, \beta_{sa} \in \mathbb{R}$ that characterize the reward distribution for each (s, a).

Algorithm 3 PS-MDP-NAS - Posterior Sampling for navigating MDPs

Require: Parameters (ρ, α, β).

Initialize counter $N_0(s, a, s') \leftarrow 0$ for all $(s, a, s') \in S \times A \times S$.
Observe $s_0 \sim p_0$.
for $t = 0, 1, 2, \ldots,$ **do**
  Computing the allocation.
    Sample a transition function $P_t(\cdot|s, a) \sim \mathrm{Dir}(\rho_{sa}(t))$ and reward distribution $q_t(\cdot|s, a) \sim \mathrm{Ber}(\alpha_{sa}(t)/(\alpha_{sa}(t) + \beta_{sa}(t)))$.
    Perform policy iteration using $\phi_t = (P_t, q_t)$ and compute $\pi_t^\star$, the greedy policy at time t. Use $\pi_t^\star$ to derive the various quantities needed to compute the allocation in Thm. 4.2
    Compute allocation $\omega^{(t)}$ by solving the optimization problem in Thm. 4.2 using (P_t, q_t).
  Sampling step.
    Sample $a_t \sim \omega^{(t)}(s_t, \cdot)$ and observe $(r_t, s_{t+1}) \sim q(\cdot|s_t, a_t) \otimes P(\cdot|s_t, a_t)$.
  Posterior update.
    Update number of visits $N_{t+1}(s_t, a_t, s_{t+1}) \leftarrow N_t(s_t, a_t, s_{t+1}) + 1$ and total cumulative reward $R_{t+1}(s_t, a_t) \leftarrow R_t(s_t, a_t) + r_t$.
    Update posterior parameters

$$\begin{aligned}\rho_{sa}(t+1) &\leftarrow \rho_{sa} + N_{t+1}(s, a, s'),\\ \alpha_{sa}(t+1) &\leftarrow \alpha_{sa} + R_{t+1}(s, a),\\ \beta_{sa}(t+1) &\leftarrow \beta_{sa} + N_{t+1}(s, a) - R_{t+1}(s, a).\end{aligned}$$

end for

After observing an experience at time t, the posterior parameters $\rho_{sa}(t), \alpha_{sa}(t), \beta_{sa}(t)$ at time t are computed as follows

$$\begin{aligned}\rho_{sa}(t) &\leftarrow \rho_{sa} + N_t(s, a, s'),\\ \alpha_{sa}(t) &\leftarrow \alpha_{sa} + R_t(s, a),\\ \beta_{sa}(t) &\leftarrow \beta_{sa} + N_t(s, a) - R_t(s, a).\end{aligned}$$

where $N_t(s, a, s')$ is the number of times the agent experienced state s' after choosing action a in state s up to time t, $R_t(s, a) = \sum_{n=0}^{t} r_n \mathbf{1}\{s_n = s \wedge a_n = a\}$ is the total cumulative reward observed up to time t after choosing action a in state s, and, lastly, $N_t(s, a) = \sum_{s'} N_t(s, a, s')$ is the total number of times the agent chose action a in state s.

B.2 O-BPI - Online Best Policy Identification

In this part, we introduce O-BPI, or Online Best Policy Identification. This procedure bears resemblance to MDP-NAS, but sidesteps the need for policy iteration at every timestep. Instead, we employ stochastic approximation to learn the (Q, M)-values and use these calculated values to compute the allocation. We describe a variation where the user exclusively learns the M-function for a given k. It's important to note, however, that the agent has the capability to learn multiple M-functions, for varying k values, to better approximate the true solution.

We present a version of the algorithm that uses forced exploration, where we mix the allocation that we obtain from Theorem 4.2 with a uniform distribution. It's straightforward to derive an extension using bootstrapping, as we show in subsequent subsections.

O-BPI. To compute the allocation ω we require to estimate the transition function (*e.g.*, using maximum likelihood), and we denote its estimate at time t by $\hat{P}_t$. To derive ω, as for MF-BPI, we compute $\pi_t^\star, \Delta_t$ and $\Delta_{\min,t}$ using the estimate Q-function Q_t. Then, we solve the following convex problem

$$\inf_{\omega \in \mathcal{C}_t} \max_{s, a \neq \pi_t^\star(s)} \frac{2 + 8\varphi^2 M_t(s, a)^{2^{1-k}}}{\omega(s, a)(\Delta_t(s, a) + \lambda)^2} + \max_{s'} \frac{C_t(s')(1+\gamma)^2}{\omega(s', \pi^\star(s'))(\Delta_t(s, a) + \lambda)^2(1-\gamma)^2}, \quad (7)$$

where $C_t(s') = \max\left(4, 16\gamma^2\varphi^2 M_t(s', \pi_t^\star(s'))^{2^{1-k}}\right)$. The constraint set $\mathcal{C}_t$ is simply $\Delta(S \times A)$ in the generative case, and $\mathcal{C}_t = \{\omega : \omega_{s'} = \sum_{s,a} \omega(s, a)\hat{P}_t(s'|s, a), \forall s'\}$ in the case with navigation

constraints. In particular, for finite state-action MDPs we use $N_t(s, a, s')$ (the number of visits up to time t of (s, a, s')) to estimate P. As in MDP-NAS [38], to ensure that the various estimates asymptotically converge to the true quantities, we force exploration using a D-tracking like procedure, that is, with probability $\epsilon_t \propto 1/N_t(s_t)^\lambda, \lambda \in (0, 1]$, we choose an action uniformly at random in state s_t at time t (since this type of forced exploration is slightly different from the one proposed in [38]. We have the following guarantee.

Lemma B.1 (Forced exploration)**.** *Let* $\epsilon_t(s) \coloneqq 1/N_t(s)^\alpha$ *with* $\alpha \in (0, 1]$*. Then,* O-BPI *satisfies* $\mathbb{P}_\phi(\forall (s, a) \in S \times A, \lim_{t\to\infty} N_t(s, a) = \infty) = 1$.

Proof. The lemma follows from Observation 1 in [59]. We use the fact that in communicating MDPs every state gets visited infinitely often as long as each action is chosen infinitely often in each state. Denote by $\mathbb{P}(a_t = a|s_t = s, N_t(s) = i)$ the probability that action a is executed at the i^{th} visit to state s. The forced exploration step in O-BPI ensures that $\mathbb{P}(a_t = a|s_t = s, N_t(s) = i) \geq \epsilon_t(s)/|A| = 1/(i^\alpha|A|)$. Consequently, for all (s, a) and $0 < \alpha \leq 1$ we have that

$$\sum_{i=1}^{\infty} \mathbb{P}(a_t = a|s_t = s, N_t(s) = i) \geq \frac{1}{|A|} \sum_{i=1}^{\infty} \frac{1}{i^\alpha} = \infty.$$

By the Borel-Cantelli lemma it follows that asymptotically each action is chosen infinitely often in each state, which yields the desired result. □

B.3 Boostrapped MF-BPI - Model Free Best Policy Identification

In this section, we describe in more detail bootstrapped MF-BPI. MF-BPI is a model-free algorithm that adapts exploration based on a sample-complexity bound, while using bootstrapping to characterize the epistemic uncertainty. The algorithm also relies on a closed-form solution for computing the allocation ω, which eliminates the need for solving an optimization problem.

Recall the closed form solution for the allocation ω from Corollary 5.1:

$$\omega(s, a) \propto \begin{cases} H(s, a) & \text{if } a \neq \pi^\star(s) \\ \sqrt{H \sum_{s, a\neq\pi^\star(s)} H(s, a)/|S|} & \text{otherwise,} \end{cases} \tag{8}$$

where $H(s, a)$ and H are defined as follows:

$$H(s, a) = \frac{2 + 8\varphi^2 M^k_{sa}[V^\star]^{2^{1-k}}}{(\Delta(s, a) + \lambda)^2}, \tag{9}$$

$$H = \frac{\max_{s'} C(s')(1 + \gamma)^2}{(\Delta_{\min} + \lambda)^2(1 - \gamma)^2}, \tag{10}$$

for some fixed value k that should be treated as a hyper-parameter, parameter $\lambda \geq 0$ and $C(s') = \max\left(4, 16\gamma^2\varphi^2 M^k_{s',\pi^\star(s')}[V^\star]^{2^{1-k}}\right)$. Rather than resorting to policy iteration, our approach involves learning the Q-values and M-values through stochastic approximation. The algorithm keeps track of estimates $Q_t(s, a)$ and $M_t(s, a)$ for all states and actions up to a given time t. The updates for the stochastic approximation are carried out at every time step t and can be represented as:

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha_t(s_t, a_t)\left(r_t + \gamma \max_a Q_t(s_{t+1}, a) - Q_t(s_t, a_t)\right), \tag{11}$$

$$M_{t+1}(s_t, a_t) = M_t(s_t, a_t) + \beta_t(s_t, a_t)\left((\delta'_t/\gamma)^{2^k} - M_t(s_t, a_t)\right). \tag{12}$$

In this equation, $\delta'_t = r_t + \gamma \max_a Q_{t+1}(s_{t+1}, a) - Q_{t+1}(s_t, a_t)$, and $(\alpha_t, \beta_t)_{t\geq 0}$ are the learning rates that meet the Robbins-Monroe conditions [13].

Bootstrap sample. Our method employs a bootstrap sampling strategy to estimate uncertainties in a non-parametric way. This approach can augment a forced exploration step, ensuring the convergence of the estimates asymptotically. We maintain a collection of (Q, M)-values and produce a new bootstrap sample $(\hat{Q}_t, \hat{M}_t)$ at every time step t. In particular, we start with an ensemble of Q-functions $Q_1, \ldots, Q_B$ (similarly for $M_1, \ldots, M_B$) initialized uniformly at random in $[0, 1/(1 - \gamma)]$

Algorithm 4 Bootstrapped MF-BPI

Require: Parameters (λ, k, p); ensemble size B; learning rates $\{(\alpha_{t,b}, \beta_{t,b})\}_{t,b}$.

1: Initialize $Q_{1,b}(s,a) \sim \mathcal{U}([0, 1/(1-\gamma)])$ and $M_{1,b}(s,a) \sim \mathcal{U}([0, 1/(1-\gamma)^{2^k}])$ for all $(s,a) \in S \times A$ and $b \in [B]$.
2: Observe $s_0 \sim p_0$.
3: **for** $t = 0, 1, 2, \ldots,$ **do**
4: **Compute allocation.**
5: Sample $\xi \sim \mathcal{U}([0,1])$ and set, $\hat{Q}_t(s,a) = \text{Quantile}_\xi(\{Q_{t,1}(s,a), \ldots, Q_{t,B}(s,a)\})$ (sim. $\hat{M}_t$) for all (s,a).
6: Compute generative solution $\omega_o^{(t)}$ using Proposition 5.1. Let $\pi_t^\star(s) = \arg\max_a \hat{Q}_t(s,a)$, $\Delta_t(s,a) = \hat{Q}_t(s, \pi_t^\star(s)) - \hat{Q}_t(s,a)$, $\Delta_{\text{min},t} = \min_{s, a \neq \pi_t^\star(s)} \Delta_t(s,a)$ and

$$H_t(s,a) \coloneqq \frac{2 + 8\varphi^2 \hat{M}_t(s,a)^{2^{1-k}}}{(\Delta_t(s,a) + \lambda)^2},$$

$$H_t \coloneqq \frac{\max_{s'} 4(1+\gamma)^2 \max(1, 4\gamma^2\varphi^2 \hat{M}_t(s', \pi_t^\star(s'))^{2^{1-k}})}{(\Delta_{\text{min},t} + \lambda)^2 (1-\gamma)^2}$$

7: Set

$$\omega_o^{(t)}(s_t, a) = \begin{cases} H_t(s_t, a) & \text{if } a \neq \pi_t^\star(s_t), \\ \sqrt{H_t \sum_{s, a \neq \pi_t^\star(s)} H_t(s,a)/|S|} & \text{otherwise .} \end{cases}$$

8: Let $\omega^{(t)}(s_t, a) = \frac{\omega_o^{(t)}(s_t, a)}{\sum_{a'} \omega_o^{(t)}(s_t, a')}$ be the policy at time t in state s_t.
9: Sample $a_t \sim \omega^{(t)}(s_t, \cdot)$; observe $(r_t, s_{t+1}) \sim q(\cdot|s_t, a_t) \otimes P(\cdot|s_t, a_t)$.
10: **Training step.**
11: **for** $b = 1, \ldots, B$ **do**
12: With probability p, using the experience (s_t, a_t, r_t, s_{t+1}), update the values $Q_{t,b}, M_{t,b}$ using Equations (5) and (6)

$$Q_{t+1,b}(s_t, a_t) = Q_{t,b}(s_t, a_t) + \alpha_{t,b}(s_t, a_t) \left(r_t + \gamma \max_a Q_{t,b}(s_{t+1}, a) - Q_{t,b}(s_t, a_t) \right),$$

$$M_{t+1,b}(s_t, a_t) = M_{t,b}(s_t, a_t) + \beta_{t,b}(s_t, a_t) \left(\left(\delta'_{t,b}/\gamma\right)^{2k} - M_{t,b}(s_t, a_t) \right),$$

where $\delta'_{t,b} = r_t + \gamma \max_a Q_{t+1,b}(s_{t+1}, a) - Q_{t+1,b}(s_t, a_t)$.
13: **end for**
14: Compute greedy policy as

$$\bar{\pi}_t^\star(s) \leftarrow \text{Median}(\{\arg\max_a Q_{t+1,1}(s,a), \ldots, \arg\max_a Q_{t+1,B}(s,a)\}).$$

15: **end for**

(similarly $[0, 1/(1-\gamma)^{2k}]$). It's important to highlight that initializing the ensemble members across the full range of potential values is essential to account for uncertainties not arising from the collected data.

At each timestep t, a bootstrap sample $\hat{Q}_t(s,a)$ (and similarly $\hat{M}_t$) is generated by sampling a uniform random variable $\xi \sim \mathcal{U}([0,1])$, and, for each (s,a), $\hat{Q}_t(s,a) = \text{Quantile}_\xi(Q_{t,1}(s,a), \ldots, Q_{t,B}(s,a))$; in other words, $\hat{Q}_t(s,a)$ is the ξ-quantile of $Q_{t,1}(s,a), \ldots, Q_{t,B}(s,a)$ assuming a linear interpolation between the Q-values. This method is akin to sampling from an empirical cumulative distribution function (CDF), where the CDF in this case embodies the uncertainty over the Q-values. We also apply the same bootstrap sampling procedure to $\hat{M}_t(s,a)$. Empirically, we found the method to work for small values of the ensemble size B for most problems, but we did not conduct extensive research on this topic. A promising venue of research is to study how the parameter p and the ensemble size B can be tuned for the problem at hand.

Computing the allocation. This bootstrap sample $(\hat{Q}_t, \hat{M}_t)$ is subsequently used to calculate the allocation $\omega^{(t)}$, where $\Delta_t(s,a) = \max_{a'} \hat{Q}_t(s,a') - \hat{Q}_t(s,a)$, $\pi_t^\star(s) = \arg\max_a \hat{Q}_t(s,a)$, and $\Delta_{\min,t} = \min_{s,a\neq\pi_t^\star(s)} \Delta_t(s,a)$. Then, we set

$$H_t(s,a) \coloneqq \frac{2 + 8\varphi^2 \hat{M}_t(s,a)^{2^{1-k}}}{(\Delta_t(s,a)+\lambda)^2},$$

$$H_t \coloneqq \frac{\max_{s'} 4(1+\gamma)^2 \max(1, 4\gamma^2\varphi^2 \hat{M}_t(s', \pi_t^\star(s'))^{2^{1-k}})}{(\Delta_{\min,t}+\lambda)^2(1-\gamma)^2}$$

as well as

$$\omega_o^{(t)}(s_t,a) = \begin{cases} H_t(s_t,a) & \text{if } a \neq \pi_t^\star(s_t), \\ \sqrt{H_t \sum_{s,a\neq\pi_t^\star(s)} H_t(s,a)/|S|} & \text{otherwise .} \end{cases}$$

The final policy is then obtain by normalizing $\omega_o^{(t)}$: $\omega^{(t)}(s_t,a) = \frac{\omega_o^{(t)}(s_t,a)}{\sum_{a'} \omega_o^{(t)}(s_t,a')}$.

Greedy policy. Lastly, an overall greedy policy $\bar{\pi}_t^\star$ can be estimated by using the ensemble of Q-functions. For example, by majority voting as

$$\bar{\pi}_t^\star(s) \leftarrow \text{Mode}(\{\arg\max_a Q_{t+1,1}(s,a), \ldots, \arg\max_a Q_{t+1,B}(s,a)\}).$$

B.4 DBMF-BPI - Deep Boostrapped Model Free Best Policy Identification

To generalize bootstrapped MF-BPI to continuous Markov Decision Processes (MDPs), we propose DBMF-BPI. DBMF-BPI leverages the concept of prior networks from BSP (Bootstrapping with Additive Prior) [50], to account for uncertainty not arising from the observed data.

Ensemble. As in the previous method, we maintain an ensemble of Q-values $Q_{\theta_1}, \ldots, Q_{\theta_B}$ (along with their target networks) and an ensemble of M-values $M_{\tau_1}, \ldots, M_{\tau_B}$. Specifically, Q-values are computed as follows for a generic b-th member of the ensemble

$$Q_{\theta_b}(s,a) = Q_{\theta_{b,0}}(s,a) + \beta_Q Q_{\theta_{b,p}}(s,a),$$

where $\beta_Q \geq 0$ is a hyper-parameter defining the scale of the prior, $\theta_{b,0}$ is a learnable parameter, and $Q_{\theta_{b,p}}$ is a fixed, randomly-initialize, Q-network that serves as a randomized prior value function. Similarly, we compute the M-values as

$$M_{\tau_b}(s,a) = M_{\tau_{b,0}}(s,a) + \beta_M M_{\tau_{b,p}}(s,a),$$

where $\beta_M \geq 0$ is a hyper-parameter, $\tau_{b,0}$ is a learnable parameter and $M_{\tau_{b,p}}$ is a fixed random prior network for the M- function.

Note that the function of the prior network is to guarantee that the (Q,M)-values are capable of covering the full spectrum of potential values. This is similar to the random initialization procedure in MF-BPI. An alternate strategy might involve initializing the network in an optimistic way (*i.e.*, by sampling parameters from a Gaussian distribution with larger variance), but our observations indicate that this may lead to worse performance.

Bootstrap sample. As before, at each timestep t a bootstrap sample $\hat{Q}_t(s,a)$ (and similarly $\hat{M}_t$) is generated by sampling a uniform random variable $\xi \sim \mathcal{U}([0,1])$, and, for each (s,a), set $\hat{Q}_t(s,a) = \text{Quantile}_\xi(Q_{t,\theta_1}(s,a), \ldots, Q_{t,\theta_B}(s,a))$. However, for numerical stability, we found it was most effective to sample ξ at the end of an episode, or every $n \propto (1-\gamma)^{-1}$ steps.

Computing the allocation. Using the bootstrap sample $(\hat{Q}_t, \hat{M}_t)$ we compute the allocation as follows. We set

$$H_t(s_t,a) = \frac{2 + 8\varphi^2 \hat{M}_t(s_t,a)^{2^{1-k}}}{(\Delta_t(s_t,a)+\lambda)^2}, \tag{13}$$

$$H_t = \frac{4(1+\gamma)^2 \max(1, 4\gamma^2\varphi^2 \hat{M}_t(s_t, \pi_t^\star(s_t))^{2^{1-k}})}{(\Delta_{\min,t}+\lambda)^2(1-\gamma)^2}, \tag{14}$$

where $\pi_t^\star(s_t) = \arg\max_a \hat{Q}_t(s_t, a)$. Note that H_t is an approximation of the true value (we are not taking the maximum over all possible states). Subsequently, we establish the allocation $\omega_o^{(t)}$: $\omega_o^{(t)}(s_t, a) = H_t(s_t, a)$ if $a \neq \pi_t^\star(s_t)$, and $\omega_o^{(t)}(s_t, a) = \sqrt{H_t \sum_{a \neq \pi_t^\star(s_t)} H_t(s_t, a)}$ otherwise. In the final step, we construct an ϵ_t-soft exploration policy $\omega^{(t)}(s_t, \cdot)$ by blending $\omega_o^{(t)}(s_t, \cdot)/\sum_a \omega_o^{(t)}(s_t, a)$ with a uniform distribution, utilizing an exploration parameter ϵ_t.

Training and minimum gap estimation. The training procedure follows that of the classical DQN algorithm [41]. Each Q-network is trained by minimizing an MSE loss criterion. We use also the MSE loss to train the M-networks over a batch sampled from the replay buffer (note that the M-networks do not require a target network).

Next, $\Delta_{\text{min},t}$ is estimated through stochastic approximation, using the smallest gap from the most recent batch of transitions retrieved from the replay buffer as a reference. In particular, the target is given by the following expression

$$\delta_t = \min_{b \in [B]} \min_{j \in \mathcal{B}} \max_{a \neq \pi_{\theta_b}^\star(s_j)} Q_{\theta_b}(s_j, \pi_{\theta_b}^\star(s_j)) - Q_{\theta_b}(s_j, a)$$

with $\pi_{\theta_b}(s) = \arg\max_a Q_{\theta_b}(s, a)$. The estimate is then updated as $\Delta_{\text{min},t+1} \leftarrow (1-\alpha_t)\Delta_{\text{min},t} + \alpha_t \delta_t$ for some learning rate $\alpha_t = O(1/t)$.

Greedy policy Lastly, an overall greedy policy $\bar{\pi}_t^\star$ can be estimated by using the ensemble of Q-functions. For example, by majority voting as

$$\bar{\pi}_t^\star(s) \leftarrow \text{Mode}(\{\arg\max_a Q_{t+1,\theta_1}(s, a), \ldots, \arg\max_a Q_{t+1,\theta_B}(s, a)\}).$$

The full pseudo-code of the algorithm can be found in the next page.

Algorithm 5 DBMF-BPI (Deep Bootstrapped Model Free BPI) - Full Algorithm

Require: Parameters (λ, k); ensemble size B; exploration rate $\{\epsilon_t\}_t$; estimate $\Delta_{\min,0}$; mask probability p.

function `MainLoop`
  Initialize replay buffer $\mathcal{D}$, networks Q_{θ_b}, M_{τ_b} and targets $Q_{\theta'_b}$ for all $b \in [B]$.
  for $t = 0, 1, 2, \ldots,$ **do**
    Sampling step.
      Compute allocation $\omega^{(t)} \leftarrow$ `ComputeAllocation`$(s_t, \{Q_{\theta_b}, M_{\tau_b}\}_{b\in[B]}, \Delta_{\min,t}, \gamma, \lambda, k, \epsilon_t)$.
      Sample $a_t \sim \omega^{(t)}(s_t, \cdot)$ and observe $(r_t, s_{t+1}) \sim q(\cdot|s_t, a_t) \otimes P(\cdot|s_t, a_t)$.
      Add transition $z_t = (s_t, a_t, r_t, s_{t+1})$ to the replay buffer $\mathcal{D}$.
    Training step.
      Sample a batch $\mathcal{B}$ from $\mathcal{D}$, and with probability p add the i^{th} experience in $\mathcal{B}$ to a sub-batch $\mathcal{B}_b$, $\forall b \in [B]$. Update the (Q, M)-values of the b^{th} member in the ensemble using $\mathcal{B}_b$: $\{Q_{\theta_b}, Q_{\theta'_b}, M_{\tau_b}\}_{b\in[B]} \leftarrow$ `Training`$(\{\mathcal{B}_b, Q_{\theta_b}, Q_{\theta'_b}, M_{\tau_b}\}_{b\in[B]})$.
      Update estimate $\Delta_{\min,t+1} \leftarrow$ `EstimateMinimumGap`$(\Delta_{\min,t}, \mathcal{B}, \{Q_{\theta_b}\}_{b\in[B]})$.
    Compute greedy policy as

$$\bar{\pi}^\star_t(s) \leftarrow \text{Median}(\{\arg\max_a Q_{t+1,\theta_1}(s,a), \ldots, \arg\max_a Q_{t+1,\theta_B}(s,a)\}).$$

  end for
end function

function `EstimateAllocation`$(s_t, \{Q_{\theta_b}, M_{\tau_b}\}_{b\in[B]}, \Delta_{\min,t}\gamma, \lambda, k, \epsilon_t)$
  Sample $\xi \sim \mathcal{U}([0,1])$ and set, $\hat{Q}_t(s_t, a) = \text{Quantile}_\xi(\{Q_{t,\theta_1}(s_t, a), \ldots, Q_{t,\theta_B}(s_t, a)\})$ (sim. $\hat{M}_t$).
  Let $\pi^\star_t(s_t) = \arg\max_a \hat{Q}_t(s_t, a)$, and set $\Delta_t(s_t, a) = \hat{Q}_t(s_t, \pi^\star_t(s_t, a)) - \hat{Q}_t(s_t, a)$.
  Compute MDP-related quantities

$$H_t(s_t, a) \coloneqq \frac{2 + 8\varphi^2 \hat{M}_t(s_t, a)^{2^{1-k}}}{(\Delta_t(s_t, a) + \lambda)^2},$$

$$H_t \coloneqq \frac{4(1+\gamma)^2 \max(1, 4\gamma^2\varphi^2 \hat{M}_t(s_t, \pi^\star_t(s_t))^{2^{1-k}})}{(\Delta_{\min,t} + \lambda)^2(1-\gamma)^2}$$

  Set

$$\omega^{(t)}_o(s_t, a) = \begin{cases} H_t(s_t, a) & \text{if } a \neq \pi^\star_t(s_t), \\ \sqrt{H_t \sum_{a \neq \pi^\star_t(s_t)} H_t(s_t, a)} & \text{otherwise .} \end{cases}$$

  Return $\omega^{(t)}(s_t, a) = \frac{\epsilon_t}{|A|} + (1-\epsilon_t)\frac{\omega^{(t)}_o(s_t,a)}{\sum_{a'} \omega^{(t)}_o(s_t,a')}$, the policy at time t in state s_t.
end function

function `Training`$(\{\mathcal{B}_b, Q_{\theta_b}, Q_{\theta'_b}, M_{\tau_b}\}_{b\in[B]})$
  for each model in the ensemble $b = 1, \ldots, B$ **do**
    Compute targets $y_j = r_j + \gamma \max_a Q_{\theta'_b}(s_{j+1}, a)$ and perform a gradient descent step on Q_{θ_b} using $\nabla_{\theta_b}(y_j - Q_{\theta_b}(s_j, a_j))^2$ for all $j \in \mathcal{B}_b$.
    Compute targets $\bar{y}_j = (r_j + \max_a Q_{\theta_b}(s_{j+1}, a) - Q_{\theta_b}(s_j, a_j))/\gamma$ and perform a gradient descent step on M_{τ_b}using $\nabla_{\tau_b}(\bar{y}_j^{2^k} - M_{\tau_b}(s_j, a_j))^2$.
  end for
  Every K steps update target models: $\theta_{b'} \leftarrow \theta_b$ for all $b \in [B]$.
  Return updated models $\{Q_{\theta_b}, Q_{\theta'_b}, M_{\tau_b}\}_{b\in[B]}$.
end function

function `EstimateMinimumGap`$(\Delta_{\min,t}, \mathcal{B}, \{Q_{\theta_b}\}_{b\in[B]})$
  Set learning rate $\alpha_t = O(1/t)$.
  Update estimate of $\Delta_{\min,t}$: let $\pi^\star_{\theta_b}(s_j) = \arg\max_a Q_{\theta_b}(s_j, a)$ and compute target

$$\delta_t = \min_{b\in[B]} \min_{j\in\mathcal{B}} \max_{a \neq \pi^\star_{\theta_b}(s_j)} Q_{\theta_b}(s_j, \pi^\star_{\theta_b}(s_j)) - Q_{\theta_b}(s_j, a)$$

  and update estimate $\Delta_{\min,t+1} \leftarrow (1-\alpha_t)\Delta_{\min,t} + \alpha_t\delta_t$.
  Return updated estimate $\Delta_{\min,t+1}$.
end function

C Proofs

In this appendix, we provide the proofs of our main results. We start with some preliminary results. We then introduce new notation to accommodate extensions beyond the assumptions made in the main body of the paper, and prove our main theorem. Specifically, we broaden our sample-complexity bounds to encompass communicating MDPs without a unique optimal policy.

C.1 Preliminaries

Let $V : \mathcal{S} \to \mathbb{R}$ be a bounded function. We show that $\mathrm{Var}_{sa}[V] \leq \mathrm{MD}_{sa}[V]^2$. This inequality follows directly from the Bhatia-Davis inequality [12]. Applied to the value function of our MDP, this result implies that in the bound derived in Theorem 4.1, the term corresponding to the span of $V^\star$ might be sometimes dominant, and we might indeed wish to remove it from the upper bound.

Lemma C.1. *Consider an MDP ϕ with $|S|$ states and a bounded vector $V \in \mathbb{R}^{|S|}$. For any (s,a), we have $\mathrm{Var}_{sa}[V] \leq \mathrm{MD}_{sa}[V]^2$. If $\mathrm{MD}_{sa}[V] \leq 1$ then $\mathrm{Var}_{sa}[V] \leq \mathrm{MD}_{sa}[V]$.*

Proof of Lemma C.1. The result is obtained leveraging the Bhatia-Davis inequality [12]. Fix (s,a), and consider a bounded vector V. Let $\mu(s,a) = \mathbb{E}_{s'\sim P(\cdot|s,a)}[V(s')]$, $M = \max_s V(s)$ and $m = \min_s V(s)$. Then, define

$$G(s,a) \coloneqq \mathbb{E}_{s'\sim P(\cdot|s,a)}[(M - V(s'))(V(s') - m)].$$

We have $G(s,a) = -mM - \mathbb{E}_{s'\sim P(\cdot|s,a)}[V(s')^2] + (M+m)\mu(s,a)$. Since $0 \leq G(s,a)$,

$$\begin{aligned}
-\mu(s,a)^2 &\leq -mM - \mathbb{E}_{s'\sim P(\cdot|s,a)}[V(s')^2] + (M+m)\mu(s,a) - \mu(s,a)^2, \\
\mathrm{Var}_{P(s,a)}[V] &\leq -mM + (M+m)\mu(s,a) - \mu(s,a)^2, \\
\mathrm{Var}_{P(s,a)}[V] &\leq (M - \mu(s,a))(\mu(s,a) - m).
\end{aligned}$$

Since $\mathrm{MD}_{sa}[V] = \|V - \mu(s,a)\|_\infty = \max(M - \mu(s,a), \mu(s,a) - m)$, we conclude that

$$\mathrm{Var}_{P(s,a)}[V] \leq \max(M - \mu(s,a), \mu(s,a) - m)^2 = \mathrm{MD}_{P(s,a)}[V]^2.$$

This also implies that, if $\mathrm{MD}_{sa}[V] \leq 1$, then $\mathrm{Var}_{sa}[V] \leq \mathrm{MD}_{sa}[V]$. □

More generally, we also note that

$$\begin{aligned}
M_{sa}^k[V^\pi]^{2^{-k}} &\leq \mathbb{E}_{s'\sim P(\cdot|s,a)}\left[\left(\max_{s'} V^\pi(s') - \mathbb{E}_{\bar{s}\sim P(\cdot|s,a)}[V^\pi(\bar{s})]\right)^{2^k}\right]^{2^{-k}}, \\
&= \mathbb{E}_{s'\sim P(\cdot|s,a)}\left[\left\|V^\pi - \mathbb{E}_{\bar{s}\sim P(\cdot|s,a)}[V^\pi(\bar{s})]\right\|_\infty^{2^k}\right]^{2^{-k}}, \\
&= \mathrm{MD}_{sa}[V^\pi].
\end{aligned}$$

C.2 Alternative upper bounds

In this subsection, we establish the alternative upper bounds $\bar{U}_\varepsilon$ of the sample complexity lower bound proposed in Theorem 4.2. Our results extend those of [37] to MDPs where the optimal policy might not be unique.

C.2.1 Sample complexity lower bound

Assume for now that the way the learner interacts with the MDP corresponds to the generative model: in each round, she can pick any (state, action) pair and observe the corresponding next state and reward. Under this model, the following theorem provides a sample complexity lower bound satisfied by any (ε,δ)-PAC algorithm.

Theorem C.2 ((δ,ε)-PAC lower bound)**.** *Consider $\varepsilon \geq 0$, and a communicating MDP ϕ, not necessarily with a unique optimal policy. Then, the sample complexity τ of any (δ,ε)-PAC algorithm under the generative model satisfies the following lower bound:*

$$\mathbb{E}_\phi[\tau] \geq T_\varepsilon \mathrm{kl}(\delta, 1-\delta), \tag{15}$$

where $T_\varepsilon = \sup_{\omega \in \Delta(S\times A)} T_\varepsilon(\omega)$ *is the optimal characteristic time, and*

$$T_\varepsilon(\omega)^{-1} = \inf_{\psi \in \mathrm{Alt}_\varepsilon(\phi)} \mathbb{E}_{(s,a)\sim\omega}[\mathrm{KL}_{\phi|\psi}(s,a)]. \tag{16}$$

The proof follows the same lines as in [37]. A similar lower bound can be derived in the forward model where the learner has to follow the system trajectory [38]: it is obtained by replacing the supremum over $\omega \in \Delta(S \times A)$ by a supremum over $\omega \in \Omega(\phi)$, to account for the navigation constraints.

C.2.2 Upper bound on $T_\varepsilon(\omega)$

As explained in [37], even for $\varepsilon = 0$, (16) is in general a non-convex problem. Therefore it may not always be possible to even approximately solve it. An alternative way, introduced in[37], consists in convexifying the problem. The solution of the new problem then gives an upper bound of T_0.

To this aim, we will start from the following result, providing a decomposition of the confusing set.

Proposition C.3. *We have* $T_\varepsilon(\omega) \le T(\omega)$ *for all* ω*, where* $T(\omega)$ *is defined as*

$$T(\omega)^{-1} = \min_{\pi \in \Pi_0^\star(\phi)} \min_{s,a\neq\pi(s)} \min_{\psi \in \mathrm{Alt}_{\pi,sa}(\phi)} \mathbb{E}_{(s,a)\sim\omega}[\mathrm{KL}_{\phi|\psi}(s,a)]. \tag{17}$$

where $\mathrm{Alt}_{\pi,sa}(\phi) = \{\psi : \phi \ll \psi, Q_\psi^\pi(s,a) > V_\psi^\pi(s)\}$.

Proof. A similar result was derived in [37]. Its proof follows directly from Lemma C.10 and Lemma C.11. From Lemma C.10 we have that the set $\mathrm{Alt}(\phi) = \{\psi : \psi \ll \phi, \Pi_0^\star(\phi) \cap \Pi_0^\star(\psi) = \emptyset\}$ contains $\mathrm{Alt}_\varepsilon(\phi)$. From Lemma C.11 we have that $\mathrm{Alt}(\phi) \subseteq \cup_{\pi\in\Pi_0^\star(\phi)} \cup_s \cup_{a\neq\pi(s)} \mathrm{Alt}_{\pi,sa}(\phi)$, where

$$\mathrm{Alt}_{\pi,sa}(\phi) = \{\psi : Q_\psi^\pi(s,a) > V_\psi^\pi(s)\}.$$

Therefore

$$T_\varepsilon(\omega)^{-1} \ge \min_{\pi \in \Pi_0^\star(\phi)} \min_{s,a\neq\pi(s)} \min_{\psi \in \mathrm{Alt}_{\pi,sa}(\phi)} \mathbb{E}_{(s,a)\sim\omega}[\mathrm{KL}_{\phi|\psi}(s,a)] = T(\omega)^{-1}.$$

□

From the previous proposition, we are able to derive the upper bound of T_ε.

Theorem C.4. *Consider a communicating MDP* ϕ*, not necessarily with a unique optimal policy. Then, for every* (s,a) *there exists* $\bar{k}(s,a) \in \mathbb{N}$ *s.t. for all* $\omega \in \Delta(S \times A)$ *we have*

$$T_\varepsilon(\omega) \le U(\omega), \tag{18}$$

with

$$U(\omega) = \max_{\pi\in\Pi_0^\star(\phi)} \max_{s,a\neq\pi(s)} \left(\frac{2+8\varphi^2 M_{sa}^{(\bar{k}(s,a))}[V_\phi^\star]^{2^{1-\bar{k}(s,a)}}}{\Delta_\pi(s,a)^2\omega(s,a)} + \max_{s'} \frac{4C^\pi(s')(1+\gamma)^2}{\omega(s',\pi(s'))\Delta_\pi(s,a)^2(1-\gamma)^2} \right), \tag{19}$$

where $\Delta_\pi(s,a) \coloneqq V_\phi^\pi(s) - Q_\phi^\pi(s,a)$ *and* $C^\pi(s') = \max\left(1, 4\gamma^2\varphi^2 M_{s'\pi(s')}^{(\bar{k}(s',\pi(s')))}[V_\phi^\star]^{2^{1-\bar{k}(s',\pi(s'))}}\right)$.

Proof. The proof is similar as that of Theorem 1 in [37]. We start from the result of Proposition C.3:

$$T_\varepsilon(\omega)^{-1} \ge \min_{\pi \in \Pi_0^\star(\phi)} \min_{s,a\neq\pi(s)} \inf_{\psi \in \mathrm{Alt}_{\pi,sa}(\phi)} \mathbb{E}_{(s,a)\sim\omega}[\mathrm{KL}_{\phi|\psi}(s,a)].$$

For a fixed (π,s,a), the constraint $\inf_{\psi\in\mathrm{Alt}_{\pi,sa}(\phi)}$ does not involve the pairs $(\tilde{s},\tilde{a}) \in S \times A \setminus \{(s,a),(s',\pi(s'))_{s'\in S}\}$. As argued in [37], by convexity, the solution must satisfy $\mathrm{KL}_{\phi|\psi}(\tilde{s},\tilde{a}) = 0$ for those pairs. Hence

$$\inf_{\psi \in \mathrm{Alt}_{\pi,sa}(\phi)} \mathbb{E}_{(s,a)\sim\omega}[\mathrm{KL}_{\phi|\psi}(s,a)] = \inf_{\psi \in \mathrm{Alt}_{\pi,sa}(\phi)} \omega(s,a)\mathrm{KL}_{\phi|\psi}(s,a) + \sum_{s'} \omega(s',\pi(s'))\mathrm{KL}_{\phi|\psi}(s',\pi(s')).$$

Let $\Delta_\pi(s,a) \coloneqq V^\pi_\phi(s) - Q^\pi_\phi(s,a)$. Then, using the fact that $Q^\pi_\psi(s,a) > V^\pi_\psi(s)$, we obtain

$$\Delta_\pi(s,a) < V^\pi_\phi(s) - Q^\pi_\phi(s,a) + Q^\pi_\psi(s,a) - V^\pi_\psi(s).$$

This is similar to condition (5) in [37]. Next, let $\Delta r(s,a) = r_\psi(s,a) - r_\phi(s,a)$, $\Delta P(s,a) = P_\psi(s,a) - P_\phi(s,a)$, where the distribution $P(s,a)$ of the next state given (s,a) is represented as a column vector of dimension $|S|$. Further define the vector difference between the value in ψ and ϕ of π: $\Delta V^\pi = \begin{bmatrix} V^\pi_\psi(s_1) - V^\pi_\phi(s_1) & \dots & V^\pi_\psi(s_{|S|}) - V^\pi_\phi(s_{|S|}) \end{bmatrix}^\top$. Then, letting $\mathbf{1}(s) = e_s$ be the unit vector with 1 in position s, we find

$$\begin{aligned}
\Delta_\pi(s,a) &< Q^\pi_\psi(s,a) - Q^\pi_\phi(s,a) - \mathbf{1}(s)^\top \Delta V^\pi, \\
&< \Delta r(s,a) + \gamma (P_\psi(s,a)^\top V^\pi_\psi - P_\phi(s,a)^\top V^\pi_\phi) - \mathbf{1}(s)^\top \Delta V^\pi, \\
&< \Delta r(s,a) + \gamma \Delta P(s,a)^\top V^\pi_\phi + (\gamma P_\psi(s,a) - \mathbf{1}(s))^\top \Delta V^\pi.
\end{aligned}$$

Now, observe that:

$$\begin{aligned}
V^\pi_\psi(s) - V^\pi_\phi(s) &= \Delta r(s,\pi(s)) + \gamma (P_\psi(s,\pi(s))^\top V^\pi_\psi - P_\phi(s,\pi(s))^\top V^\pi_\phi), \\
&= \Delta r(s,\pi(s)) + \gamma (P_\psi(s,\pi(s))^\top \Delta V^\pi + \Delta P(s,\pi(s))^\top V^\pi_\phi), \\
&\leq \left| \Delta r(s,\pi(s)) + \gamma (P_\psi(s,\pi(s))^\top \Delta V^\pi + \Delta P(s,\pi(s))^\top V^\pi_\phi) \right|, \\
&\leq \max_{s'} |\Delta r(s',\pi(s')) + \gamma \Delta P(s',\pi(s'))^\top V^\pi_\phi| + \gamma \max_{\tilde{s}} |V^\pi_\psi(\tilde{s}) - V^\pi_\phi(\tilde{s})|.
\end{aligned}$$

We deduce that:

$$\|\Delta V^\pi\|_\infty \leq \frac{1}{1-\gamma} \left[\max_{s'} |\Delta r(s',\pi(s'))| + \gamma |\Delta P(s',\pi(s'))^\top V^\pi_\phi| \right].$$

Using the fact that $\|\gamma P_\psi(s,a) - \mathbf{1}(s)\|_1 = |\gamma P(s|s,a) - 1| + \gamma(1 - P(s|s,a)) \leq 1 + \gamma$, we can bound $|(\gamma P_\psi(s,a) - \mathbf{1}(s))^\top \Delta V^\pi|$ as follows:

$$\begin{aligned}
|(\gamma P_\psi(s,a) - \mathbf{1}(s))^\top \Delta V^\pi| &\leq \|\gamma P_\psi(s,a) - \mathbf{1}(s)\|_1 \|\Delta V^\pi\|_\infty \\
&\leq \frac{1+\gamma}{1-\gamma} \left[\max_{s'} |\Delta r(s',\pi(s'))| + \gamma |\Delta P(s',\pi(s'))^\top V^\pi_\phi| \right].
\end{aligned}$$

Therefore,

$$\Delta_\pi(s,a) < |\Delta r(s,a)| + \gamma |\Delta P(s,a)^\top V^\pi_\phi| + \tfrac{1+\gamma}{1-\gamma} \left[\max_{s'} |\Delta r(s',\pi(s'))| + \gamma |\Delta P(s',\pi(s'))^\top V^\pi_\phi| \right].$$

Write each of the terms as a fraction of $\Delta_\pi(s,a)$ using $\{\alpha_i\}_{i=1}^3$, which are non-negative terms satisfying $\sum_{i=1}^3 \alpha_i > 1$:

$$\begin{cases}
\alpha_1 \Delta_\pi(s,a) = |\Delta r(s,a)|, \\
\alpha_2 \Delta_\pi(s,a) = \gamma |\Delta P(s,a)^\top V^\pi_\phi|, \\
\alpha_3 \Delta_\pi(s,a) = \dfrac{1+\gamma}{1-\gamma} \max_{s'} \left[|\Delta r(s',\pi(s'))| + \gamma |\Delta P(s',\pi(s'))^\top V^\pi_\phi| \right].
\end{cases}$$

For the first term, using the Pinsker inequality, we immediately get: $(\alpha_1 \Delta_\pi(s,a))^2 \leq 2\mathrm{KL}_{q_\phi, q_\psi}(s,a)$.

For the second term, using Lemma C.9, we obtain:

$$(\alpha_2 \Delta_\pi(s,a))^2 \leq 8\gamma^2 \varphi^2 M_{sa}^{(\bar{k}(s,a))} [V^\star_\phi]^{2^{1-\bar{k}(s,a)}} \mathrm{KL}_{P_\phi, P_\psi}(s,a).$$

Finally, to bound the last term, using $(a+b)^2 \leq 2(a^2+b^2)$, Lemma C.9 and the Pinsker inequality, we have

$$\begin{aligned}
\left(|\Delta r(s',\pi(s'))| + \gamma |\Delta P(s',\pi(s'))^\top V^\pi_\phi| \right)^2 &\leq 2 \left(|\Delta r(s',\pi(s'))|^2 + \gamma^2 |\Delta P(s',\pi(s'))^\top V^\pi_\phi|^2 \right), \\
&\leq 2 \Big(2\mathrm{KL}_{q_\phi, q_\psi}(s',\pi(s')) + 8\gamma^2 \varphi^2 M_{s'\pi(s')}^{(\bar{k}(s',\pi(s')))} [V^\star_\phi]^{2^{1-\bar{k}(s',\pi(s'))}} \mathrm{KL}_{P_\phi, P_\psi}(s',\pi(s')) \Big), \\
&\leq 4C^\pi(s') (\mathrm{KL}_{q_\phi, q_\psi}(s',\pi(s')) + \mathrm{KL}_{P_\phi, P_\psi}(s',\pi(s'))),
\end{aligned}$$

with $C^\pi(s') = \max\left(1, 4\gamma^2\varphi^2 M_{s'\pi(s')}^{(\bar{k}(s',\pi(s')))}[V_\phi^\star]^{2^{1-\bar{k}(s',\pi(s'))}}\right)$. Therefore

$$\begin{aligned}
\alpha_3^2\frac{(1-\gamma)^2}{(1+\gamma)^2}\Delta_\pi(s,a)^2 &\leq 4\max_{s'} C^\pi(s')(\mathrm{KL}_{q_\phi,q_\psi}(s',\pi(s')) + \mathrm{KL}_{P_\phi,P_\psi}(s',\pi(s'))),\\
&= 4\max_{s'}\frac{\omega(s',\pi(s'))}{\omega(s',\pi(s'))}C^\pi(s')(\mathrm{KL}_{q_\phi,q_\psi}(s',\pi(s')) + \mathrm{KL}_{P_\phi,P_\psi}(s',\pi(s'))),\\
&\leq 4\max_{\tilde{s}}\frac{C^\pi(\tilde{s})}{\omega(\tilde{s},\pi(\tilde{s}))}\max_{s'}\omega(s',\pi(s'))(\mathrm{KL}_{q_\phi,q_\psi}(s',\pi(s')) + \mathrm{KL}_{P_\phi,P_\psi}(s',\pi(s'))).
\end{aligned}$$

In conclusion, we have the following set of inequalities:

$$\begin{aligned}
\frac{\omega(s,a)(\alpha_1\Delta_\pi(s,a))^2}{2} &\leq \omega(s,a)\mathrm{KL}_{q_\phi,q_\psi}(s,a),\\
\frac{\omega(s,a)(\alpha_2\Delta_\pi(s,a))^2}{8\gamma^2\varphi^2 M_{P_\phi(s,a)}^{(\bar{k}(s,a))}[V_\phi^\star]^{2^{1-\bar{k}(s,a)}}} &\leq \omega(s,a)\mathrm{KL}_{P_\phi,P_\psi}(s,a),\\
\min_{s'}\frac{\omega(s',\pi(s'))(\alpha_3(1-\gamma)\Delta_\pi(s,a))^2}{4C^\pi(s')(1+\gamma)^2} &\leq \max_{s'}\omega(s',\pi(s'))(\mathrm{KL}_{q_\phi,q_\psi}(s',\pi(s'))\\
&\quad + \mathrm{KL}_{P_\phi,P_\psi}(s',\pi(s'))).
\end{aligned}$$

As in [37] we observe that we can replace α_i by $\alpha_i/\sum_j\alpha_j$ (since $\sum_i\alpha_i > 1$). Consequently, denoting by Δ_n the n-dimensional simplex, we have

$$\begin{aligned}
T_\varepsilon(\omega)^{-1} &\geq \min_{\pi\in\Pi_0^\star(\phi)}\min_{s,a\neq\pi(s)}\inf_{\psi\in\bar{\mathrm{Alt}}_{\pi,sa,\varepsilon}(\phi)}\omega(s,a)\mathrm{KL}_{\phi|\psi}(s,a) + \sum_{s'}\omega(s',\pi(s'))\mathrm{KL}_{\phi|\psi}(s',\pi(s')).\\
&\geq \min_{\pi\in\Pi_0^\star(\phi)}\min_{s,a\neq\pi(s)}\inf_{\alpha\in\Delta_3}\sum_{i=1}^3 B_i(s,a)\alpha_i^2.
\end{aligned}$$

where

$$\begin{aligned}
B_1(s,a) &= \omega(s,a)\Delta_\pi(s,a)^2/2,\\
B_2(s,a) &= \omega(s,a)\frac{\Delta_\pi(s,a)^2}{8\gamma^2\varphi^2 M_{sa}^{(\bar{k}(s,a))}[V_\phi^\star]^{2^{1-\bar{k}(s,a)}}},\\
B_3(s,a) &= \min_{s'}\omega(s',\pi(s'))\frac{(\Delta_\pi(s,a)(1-\gamma))^2}{4C^\pi(s')(1+\gamma)^2}.
\end{aligned}$$

Therefore $T_\varepsilon(\omega)^{-1} \geq \min_{\pi\in\Pi_0^\star(\phi)}\min_{s,a\neq\pi(s)}\left(\sum_{i=1}^3 B_i(s,a)^{-1}\right)^{-1}$, from which we conclude that:

$$T_\varepsilon(\omega) \leq \max_{\pi\in\Pi_0^\star(\phi)}\max_{s,a\neq\pi(s)}\left(\frac{2+8\gamma^2\varphi^2 M_{sa}^{(\bar{k}(s,a))}[V_\phi^\star]^{2^{1-\bar{k}(s,a)}}}{\Delta_\pi(s,a)^2\omega(s,a)} + \max_{s'}\frac{4C^\pi(s')(1+\gamma)^2}{\omega(s',\pi(s'))\Delta_\pi(s,a)^2(1-\gamma)^2}\right). \tag{20}$$

□

C.2.3 Closed form solution under the generative model

Under the generative model, we are able to find a closed-form solution of the sample complexity upper bound by slightly relaxing our upper bound of $T_\varepsilon(\omega)$. The procedure is similar to that used in [37].

Theorem C.5. *Let $\varepsilon \geq 0$, and a communicating MDP ϕ, with a unique optimal policy π^star. Then, for all $\omega \in \Delta(S\times A)$, we have:*

$$T_\varepsilon(\omega) \leq U(\omega) \leq \tilde{U}(\omega), \tag{21}$$

where $U(\omega)$ is defined in the previous theorem, and

$$\tilde{U}(\omega) = \max_{s,a\neq\pi^\star(s)}\frac{2+8\varphi^2 M_{sa}^{(\bar{k}(s,a))}[V_\phi^\star]^{2^{1-\bar{k}(s,a)}}}{\Delta(s,a)^2\omega(s,a)} + \frac{\max_{s'}4C^{\pi^\star}(s')(1+\gamma)^2}{\min_{\tilde{s}}\omega(\tilde{s},\pi^\star(\tilde{s}))\Delta_{\min}^2(1-\gamma)^2}. \tag{22}$$

where $\Delta(s,a) \coloneqq V_\phi^{\pi^\star}(s) - Q_\phi^{\pi^\star}(s,a)$.

Proof. The proof follows from the previous theorem. Since there is a unique optimal policy we have $\Delta_\pi(s,a) \geq \Delta_{\min}$, and thus

$$\begin{aligned}\tilde{U}(\omega) &\leq \max_{s,a\neq\pi^\star(s)} \frac{2+8\varphi^2 M_{sa}^{(\bar{k}(s,a))}[V_\phi^\star]^{2^{1-\bar{k}(s,a)}}}{\Delta(s,a)^2\omega(s,a)} + \max_{s'} \frac{4C^{\pi^\star}(s')(1+\gamma)^2}{\omega(s',\pi^\star(s'))\Delta_{\min}^2(1-\gamma)^2}, \\ &\leq \max_{s,a\neq\pi^\star(s)} \frac{2+8\varphi^2 M_{sa}^{(\bar{k}(s,a))}[V_\phi^\star]^{2^{1-\bar{k}(s,a)}}}{\Delta(s,a)^2\omega(s,a)} + \frac{\max_{s'} 4C^{\pi^\star}(s')(1+\gamma)^2}{\min_{\tilde{s}}\omega(\tilde{s},\pi^\star(\tilde{s}))\Delta_{\min}^2(1-\gamma)^2}.\end{aligned}$$

□

For this particular bound, as in [37], we are able to find a closed form expression of the optimal generative allocation $\omega^\star \in \arg\min_{\omega\in\Delta(S\times A)} \tilde{U}(\omega)$ leading to an upper bound of the sample complexity lower bound. The following corollary is obtained by simply solving the optimization problem $\inf_{\omega\in\Delta(S\times A)} \tilde{U}(\omega)$.

Corollary C.6. *Consider a communicating MDP with unique optimal policy. Consider the bound defined in the previous theorem by $\tilde{U}(\omega)$. Then, the generative solution $\omega^\star = \arg\inf_{\omega\in\Delta(S\times A)} \tilde{U}(\omega)$ is given by*

$$\omega(s,a) = \begin{cases} H(s,a)/\Gamma & s,a \neq \pi^\star(s), \\ \sqrt{H\sum_{s,a\neq\pi^\star(s)} H(s,a)/|S|}/\Gamma & \textit{otherwise}. \end{cases} \tag{23}$$

where

$$H(s,a) = \frac{2+8\varphi^2 M_{sa}^{(\bar{k}(s,a))}[V_\phi^\star]^{2^{1-\bar{k}(s,a)}}}{\Delta(s,a)^2\omega(s,a)}, \quad H = \max_{s'} \frac{4C^{\pi^\star}(s')(1+\gamma)^2}{\Delta_{\min}^2(1-\gamma)^2}, \tag{24}$$

$$\Gamma = \sum_{s,a\neq\pi^\star(s)} H(s,a) + \sqrt{|S|H \sum_{s,a\neq\pi^\star(s)} H(s,a)}. \tag{25}$$

Furthermore, the value of the problem is:

$$\inf_{\omega\in\Delta(S\times A)} \tilde{U}(\omega) = \left(\sqrt{\sum_{s,a\neq\pi^\star(s)} H(s,a)} + \sqrt{|S|H}\right)^2 \leq 2\left(\sum_{s,a\neq\pi^\star(s)} H(s,a) + |S|H\right). \tag{26}$$

C.2.4 Technical lemmas

We finally state and prove the lemmas used in the derivation of our upper bounds of the sample complexity lower bound. These lemmas can be seen as an alternative to Lemma 4 used by the authors of [37] to derive their upper bounds.

In what follows, we consider a finite set $\Omega = \{\omega_1, \dots, \omega_N\}$. For each $\omega \in \Omega$, let $f(\omega)$ be a real number, and we define the vector $\mathbf{f}(\Omega) = [f(\omega_1) \quad \dots \quad f(\omega_N)]^\top$.

We start by a result, that can be deducted from the proof of Lemma 4 in [37].

Lemma C.7. *Let P, Q be pmfs over some finite space $\Omega = \{\omega_1, \dots, \omega_N\}$. Let $f : \Omega \to \mathbb{R}$ and $\mathbf{f}(\Omega) \coloneqq [f(\omega_1) \quad \dots \quad f(\omega_N)]^\top$.*
Finally, we introduce the elementwise power[2] *$\mathbf{f}^{\circ k}(\Omega) = \begin{bmatrix} f(\omega_1)^k & \dots & f(\omega_N)^k \end{bmatrix}^\top$. Then*

$$|(P-Q)^\top \mathbf{f}(\Omega)|^2 \leq 4d_H(P,Q)^2 \Big(2\mathbb{E}_{\omega\sim Q}[f(\omega)^2] + (P-Q)^\top(\mathbf{f}^{\circ 2}(\Omega))\Big), \tag{27}$$

where d_H is the Hellinger distance.

[2] also known as as Hadamard power.

Proof. The proof can be easily deduced from Lemma 4 in [37]. We present the proof for completeness. Let $\sqrt{P}$ be the square root of the elements in P (sim. $\sqrt{Q}$). We have:

$$\begin{aligned}(P-Q)^\top \mathbf{f}(\Omega) &= \sum_\omega (P(\omega)-Q(\omega))f(\omega),\\ &= \sum_\omega (\sqrt{P(\omega)}-\sqrt{Q(\omega)})(\sqrt{P(\omega)}+\sqrt{Q(\omega)})f(\omega),\\ &= (\sqrt{P}-\sqrt{Q})^\top[(\sqrt{P}+\sqrt{Q})\circ \mathbf{f}(\Omega)],\end{aligned}$$

where $\circ$ is the Hadamard product. We apply the Cauchy-Schwartz inequality to the last term to get:

$$|(P-Q)^\top \mathbf{f}(\Omega)|^2 \leq \|\sqrt{P}-\sqrt{Q}\|_2^2 \|(\sqrt{P}+\sqrt{Q})\circ \mathbf{f}(\Omega)\|_2^2.$$

Note that $\|\sqrt{P}-\sqrt{Q}\|_2 = \sqrt{2}d_H(P,Q)$. Regarding $\|(\sqrt{P}+\sqrt{Q})\circ \mathbf{f}(\Omega)\|_2$, using the inequality $(a+b)^2 \leq 2(a^2+b^2)$, we have:

$$\begin{aligned}\|(\sqrt{P}+\sqrt{Q})\circ \mathbf{f}(\Omega)\|_2^2 &\leq 2\sum_\omega (P(\omega)+Q(\omega))f(\omega)^2,\\ &= 2\sum_\omega (2Q(\omega)+P(\omega)-Q(\omega))f(\omega)^2,\\ &= 2\left(2\mathbb{E}_{\omega\sim Q}[f(\omega)^2] + (P-Q)^\top \mathbf{f}^{\circ 2}(\Omega)\right),\end{aligned}$$

which concludes the proof. □

Applying the above lemma recursively, we obtain the following result.

Lemma C.8. *Consider $f:\Omega\to\mathbb{R}$ as before. Assume that $\max_{\omega\in\Omega}|f(\omega)| \leq F < \infty$. Then,*

$$|(P-Q)^\top \mathbf{f}(\Omega)| \leq \sqrt{8}\varphi d_H(P,Q) \sup_{k\geq 1} \mathbb{E}_{\omega\sim Q}[f(\omega)^{2^k}]^{2^{-k}}, \tag{28}$$

where φ is the golden ratio.

Proof. The idea is to observe that we can use Lemma C.7 to bound $(P-Q)^\top \mathbf{f}^{\circ 2}(\Omega)$ in Equation (27). Then

$$|(P-Q)^\top \mathbf{f}^{\circ k}(\Omega)|^2 \leq 4d_H(P,Q)^2\Big(2\mathbb{E}_{\omega\sim Q}[f(\omega)^{2k}] + (P-Q)^\top \mathbf{f}^{\circ 2k}(\Omega)\Big).$$

For brevity, let $M_k = \mathbb{E}_{\omega\sim Q}[f(\omega)^k]$, then

$$\begin{aligned}&|(P-Q)^\top \mathbf{f}(\Omega)| \leq 2d_H(P,Q)\sqrt{2M_2 + (P-Q)^\top \mathbf{f}^{\circ 2}(\Omega)},\\ &\leq 2d_H(P,Q)\sqrt{2M_2 + 2d_H(P,Q)\sqrt{2M_4 + (P-Q)^\top \mathbf{f}^{\circ 4}(\Omega)}},\\ &\leq \alpha\sqrt{2M_2 + \alpha\sqrt{2M_4 + \alpha\sqrt{2M_8+\cdots}}},\end{aligned}$$

where $\alpha = 2d_H(P,Q)$. A further rewriting yields

$$\begin{aligned}&\alpha\sqrt{2M_2 + \alpha\sqrt{2M_4 + \alpha\sqrt{2M_8+\cdots}}},\\ &= \sqrt{2\alpha^2 M_2 + \alpha^3\sqrt{2M_4 + \alpha\sqrt{2M_8+\cdots}}},\\ &= \sqrt{2\alpha^2 M_2 + \sqrt{2\alpha^6 M_4 + \alpha^7\sqrt{2M_8+\cdots}}},\\ &= \sqrt{2\alpha^2 M_2 + \sqrt{2\alpha^6 M_4 + \sqrt{2\alpha^{14} M_8+\cdots}}},\end{aligned}$$

and note that the k-th term is given by $a_k = 2\alpha^{2(2^k-1)}M_{2^k}$. Consider now the sequence $b_k = (a_k)^{2^{-k}}$, and note that

$$\sup_{k\geq 1} b_k \leq \sup_{k\geq 1} \underbrace{\left(2\alpha^{2(2^k-1)}\right)^{2^{-k}}}_{(\bullet)} \cdot \sup_{k\geq 1} M_{2^k}^{2^{-k}}.$$

Observe that $(\bullet) = 2^{2^{-k}}\alpha^{2-2^{-k+1}}$ is a positive decreasing sequence, therefore we have that $\sup_{k\geq 1} b_k \leq \alpha\sqrt{2}\cdot \sup_{k\geq 1} M_{2^k}^{2^{-k}}$.

Now, we notice that $M_{2^k}^{2^{-k}}$ is bounded for all $k \geq 1$ from the boundedness of f over Ω

$$M_{2^k}^{2^{-k}} = \mathbb{E}_\omega[f(\omega)^{2^k}]^{2^{-k}} \leq F < \infty.$$

Hence, by letting $M = \alpha\sqrt{2}\cdot \sup_{k\geq 1} M_{2^k}^{2^{-k}}$, and using Herschfeld's convergence theorem [26], we find the desired result:

$$\begin{aligned}
&\sqrt{2\alpha^2 M_2 + \sqrt{2\alpha^6 M_4 + \sqrt{2\alpha^{14} M_8 + \cdots}}} \\
&\leq \sqrt{M^2 + \sqrt{M^{2^2} + \sqrt{M^{2^3} + \cdots}}}, \\
&= M\sqrt{1 + \sqrt{1 + \sqrt{1 + \cdots}}} = M\varphi.
\end{aligned}$$

□

We are now ready to state the result that serves as an alternative to Lemma 4 in [37]. Let $(\Delta P(s,a))_{s'} = P_\psi(s'|s,a) - P_\phi(s'|s,a)$.

Lemma C.9. *Consider a fixed state-action pair (s,a) and define $\bar{V}_\phi^\pi(s,a) \coloneqq \mathbb{E}_{s'\sim P_\phi(\cdot|s,a)}[V_\phi^\pi(s')]$. Let $f_\phi^\pi(s,a,s') = V_\phi^\pi(s') - \bar{V}_\phi^\pi(s,a)$ and $M_k(s,a) = \mathbb{E}_{s'\sim P_\phi(\cdot|s,a)}[f_\phi^\pi(s,a,s')^{2^k}]$. Then, there exists $\bar{k} \in \mathbb{N}$ such that*

$$|\Delta P(s,a)^\top \mathbf{f}_\phi^\pi(s,a)|^2 \leq 8\varphi^2 \mathrm{KL}_{P_\phi,P_\psi}(s,a) M_{\bar{k}}(s,a)^{2^{1-\bar{k}}}, \tag{29}$$

where $\mathbf{f}_\phi^\pi(s,a) = \begin{bmatrix} f_\phi^\pi(s,a,s_1) & f_\phi^\pi(s,a,s_2) & \cdots & f_\phi^\pi(s,a,s_{|S|}) \end{bmatrix}^\top$ *and* $\mathrm{KL}_{P_\phi,P_\psi}(s,a) = \mathrm{KL}(P_\phi(s,a), P_\psi(s,a))$.

Proof. Consider a fixed (s,a). For any $s' \in S$ we have that $|f_\phi^\pi(s,a,s')| \leq \mathrm{MD}_{sa}[V_\phi^\pi]$, therefore $\|\mathbf{f}_\phi^\pi(s,a)\|_\infty < \infty$. Using Lemma C.8 with $\mathbf{f}_\phi^\pi(s,a)$ we find the result by taking the square on both sides, and using that $d_H^2(P,Q) \leq \mathrm{KL}(P,Q)$.

□

C.2.5 Decomposition of the set of confusing MDPs

Decomposing the set $\mathrm{Alt}_\varepsilon(\phi)$ directly presents several challenges. It does even seem possible to obtain a decomposition easy to work with. Instead, we relax the problem and work on $\mathrm{Alt}(\phi) = \{\psi : \psi \ll \phi, \Pi_0^\star(\phi) \cap \Pi_0^\star(\psi) = \emptyset\}$, a set containing $\mathrm{Alt}_\varepsilon(\phi)$.

Lemma C.10. *Let $\varepsilon \geq 0$. Then, in general $\mathrm{Alt}_\varepsilon(\phi) \subseteq \mathrm{Alt}(\phi)$.*

Proof. The statement can be derived by contradiction: assume that $\psi \in \mathrm{Alt}_\varepsilon(\phi)$ does not belong to $\mathrm{Alt}(\phi)$. However, that implies that there is $\pi \in \Pi_0^\star(\phi)$ s.t. $\pi \in \Pi_0^\star(\psi)$, which is not true since by assumption $\Pi_\varepsilon^\star(\phi) \cap \Pi_\varepsilon^\star(\psi) = \emptyset$. □

Lemma C.11. *Let $\mathrm{Alt}(\phi) = \{\psi : \psi \ll \phi, \Pi_0^\star(\phi) \cap \Pi_0^\star(\psi) = \emptyset\}$. Then $\mathrm{Alt}(\phi) \subseteq \cup_{\pi\in\Pi_0^\star(\phi)} \cup_s \cup_{a\neq\pi(s)} \mathrm{Alt}_{\pi,sa}(\phi)$, where*

$$\mathrm{Alt}_{\pi,sa}(\phi) = \{\psi : \psi \ll \phi, Q_\psi^\pi(s,a) > V_\psi^\pi(s)\}.$$

Proof. The proof follows the same steps as that of the decomposition lemma in [37], and we give it for completeness.

By contradiction, consider $\psi \in \mathrm{Alt}(\phi)$ s.t. for all $\pi \in \Pi_0^\star(\phi)$ and $s, a \neq \pi(s)$ we have $Q_\psi^\pi(s,a) \leq V_\psi^\pi(s)$. Since $Q_\psi^\pi(s,\pi(s)) = V_\psi^\pi(s)$, the following inequality holds for all $\pi \in \Pi_0^\star(\phi)$ and for all (s,a)

$$Q_\psi^\pi(s,a) \leq V_\psi^\pi(s).$$

Define the Bellman operator for a generic policy π' under ψ as $(T_\psi^{\pi'} V)(s) = r_\psi(r, \pi'(s)) + \mathbb{E}_{s' \sim P(s,\pi'(s))}[V(s')]$. Then, from the above inequality that holds for all (s,a) we get the following result

$$T_\psi^{\pi_\psi^\star} V_\psi^\pi \leq V_\psi^\pi.$$

By monotonicity of the Bellman operator, we get $T_\psi^{\pi_\psi^\star} T_\psi^{\pi_\psi^\star} V \leq T_\psi^{\pi_\psi^\star} V_\psi^\pi \leq V_\psi^\pi$. Iterating, we find

$$V_\psi^{\pi_\psi^\star} = \lim_{n \to \infty} \left(T_\psi^{\pi_\psi^\star}\right)^n V \leq V_\psi^\pi,$$

which is a contradiction since π is not optimal under ψ.

□