Safe Policy Improvement with Soft Baseline Bootstrapping

Kimia Nadjahi[1†*(✉)], Romain Laroche[2*(✉)], and Rémi Tachet des Combes[2]

[1] LTCI, Télécom Paris, Institut Polytechnique de Paris, France
kimia.nadjahi@telecom-paris.fr
[2] Microsoft Research Montréal, Canada
{romain.laroche, remi.tachet}@microsoft.com.

Abstract. Batch Reinforcement Learning (Batch RL) consists in training a policy using trajectories collected with another policy, called the behavioural policy. Safe policy improvement (SPI) provides guarantees with high probability that the trained policy performs better than the behavioural policy, also called baseline in this setting. Previous work shows that the SPI objective improves mean performance as compared to using the basic RL objective, which boils down to solving the MDP with maximum likelihood (Laroche et al., 2019). Here, we build on that work and improve more precisely the SPI with Baseline Bootstrapping algorithm (SPIBB) by allowing the policy search over a wider set of policies. Instead of binarily classifying the state-action pairs into two sets (the *uncertain* and the *safe-to-train-on* ones), we adopt a softer strategy that controls the error in the value estimates by constraining the policy change according to the local model uncertainty. The method can take more risks on uncertain actions all the while remaining provably-safe, and is therefore less conservative than the state-of-the-art methods. We propose two algorithms (one optimal and one approximate) to solve this constrained optimization problem and empirically show a significant improvement over existing SPI algorithms both on finite MDPS and on infinite MDPs with a neural network function approximation.

1 Introduction

In sequential decision-making problems, a common goal is to find a good policy using a limited number of trajectories generated by another policy, usually called the behavioral policy. This approach, also known as Batch Reinforcement Learning (Lange et al., 2012), is motivated by the many real-world applications that naturally fit a setting where data collection and optimization are decoupled (contrary to online learning which integrates the two): *e.g.* dialogue systems (Singh et al., 1999), technical process control (Ernst et al., 2005; Riedmiller, 2005), medical applications (Guez et al., 2008).

While most reinforcement learning techniques aim at finding a high-performance policy (Sutton and Barto, 1998), the final policy does not necessarily perform well once

*Equal contribution.
†Work done while interning at Microsoft Research Montréal.
Finite MDPs code available at https://github.com/RomainLaroche/SPIBB.
SPIBB-DQN code available at https://github.com/rems75/SPIBB-DQN.

it is deployed. In this paper, we focus on Safe Policy Improvement (SPI, Thomas, 2015; Petrik et al., 2016), where the goal is to train a policy on a batch of data and guarantee with high probability that it performs at least as well as the behavioural policy, called baseline in this SPI setting. The safety guarantee is crucial in real-world applications where bad decisions may lead to harmful consequences.

Among the existing SPI algorithms, a recent computationally efficient and provably-safe methodology is SPI with Baseline Bootstrapping (SPIBB, Laroche et al., 2019; Simão and Spaan, 2019). Its principle consists in building the set of state-action pairs that are only encountered a few times in the dataset. This set is called the bootstrapped set. The algorithm then reproduces the baseline policy for all pairs in that set and trains greedily on the rest. It therefore assumes access to the baseline policy, which is a common assumption in the SPI literature (Petrik et al., 2016). Other SPI algorithms use as reference the baseline performance, which is assumed to be known instead (Thomas, 2015; Petrik et al., 2016). We believe that the known policy assumption is both more informative and more common, since most Batch RL settings involve datasets that were collected using a previous system based on a previous algorithm (*e.g.* dialogue, robotics, pharmaceutical treatment). While the empirical results show that SPIBB is safe and performs significantly better than the existing algorithms, it remains limited by the binary classification of the bootstrapped set: a pair either belongs to it, and the policy cannot be changed, or it does not, and the policy can be changed entirely.

Our contribution is a reformulation of the SPIBB objective that allows slight policy changes for uncertain state-action pairs while remaining safe. Instead of binarily classifying the state-action pairs into two sets, the uncertain and the safe-to-train-on ones, we adopt a strategy that extends the policy search to soft policy changes, which are constrained by an error bound related to the model uncertainty. The method is allowed to take more risks than SPIBB on uncertain actions, and still has theoretical safety guarantees under some assumptions. As a consequence, the safety constraint is softer: we coin this new SPI methodology *Safe Policy Improvement with Soft Baseline Bootstrapping* (Soft-SPIBB). We develop two algorithms to tackle the Soft-SPIBB problem. The first one solves it exactly, but is computationally expensive. The second one provides an approximate solution but is much more efficient computation-wise. We empirically evaluate the performance and safety of our algorithms on a gridworld task and analyze the reasons behind their significant advantages over the competing Batch RL algorithms. We further demonstrate the tractability of the approach by designing a DQN algorithm enforcing the Soft-SPIBB constrained policy optimization. The empirical results, obtained on a navigation task, show that Soft-SPIBB safely improves the baseline, and again outperforms all competing algorithms.

2 Background

2.1 Markov Decision Processes

We consider problems in which the agent interacts with an environment modeled as a *Markov Decision Process* (MDP): $M^* = \langle \mathcal{X}, \mathcal{A}, P^*, R^*, \gamma \rangle$, where $\mathcal{X}$ is the set of states, $\mathcal{A}$ the set of actions, P^* the unknown transition probability function, R^*

the unknown stochastic reward function bounded by $\pm R_{max}$, and $\gamma \in [0, 1)$ the discount factor for future rewards. The goal is to find a policy $\pi : \mathcal{X} \to \Delta_{\mathcal{A}}$, with $\Delta_{\mathcal{A}}$ the set of probability distributions over the set of actions $\mathcal{A}$, that maximizes the expected return of trajectories $\rho(\pi, M^*) = V^{\pi}_{M^*}(x_0) = \mathbb{E}_{\pi, M^*}\left[\sum_{t\geq 0} \gamma^t R^*(x_t, a_t)\right]$. x_0 is the initial state of the environment and $V^{\pi}_{M^*}(x)$ is the value of being in state x when following policy π in MDP M^*. We denote by Π the set of stochastic policies. Similarly to $V^{\pi}_{M^*}(x)$, $Q^{\pi}_{M^*}(x, a)$ denotes the value of taking action a in state x. $A^{\pi}_{M}(x, a) = Q^{\pi}_{M}(x, a) - V^{\pi}_{M}(x)$ quantifies the advantage (or disadvantage) of action a in state x.

Given a dataset of transitions $\mathcal{D} = \langle x_j, a_j, r_j, x'_j \rangle_{j \in [\![1, |\mathcal{D}|]\!]}$, we denote the state-action pair counts by $N_{\mathcal{D}}(x, a)$, and its Maximum Likelihood Estimator (MLE) MDP by $\widehat{M} = \langle \mathcal{X}, \mathcal{A}, \widehat{P}, \widehat{R}, \gamma \rangle$, with:

$$\widehat{P}(x'|x, a) = \frac{\sum_{\langle x_j = x, a_j = a, r_j, x'_j = x' \rangle \in \mathcal{D}} 1}{N_{\mathcal{D}}(x, a)} \text{ and } \widehat{R}(x, a) = \frac{\sum_{\langle x_j = x, a_j = a, r_j, x'_j \rangle \in \mathcal{D}} r_j}{N_{\mathcal{D}}(x, a)}.$$

The difference between an estimated parameter and the true one can be bounded using classic concentration bounds applied to the state-action counts in $\mathcal{D}$ (Petrik et al., 2016; Laroche et al., 2019): for all state-action pairs (x, a), we know with probability at least $1 - \delta$ that,

$$||P^*(\cdot|x, a) - \widehat{P}(\cdot|x, a)||_1 \leq e_P(x, a), \ |R^*(x, a) - \widehat{R}(x, a)| \leq e_P(x, a) R_{max}, \quad (1)$$

$$\left|Q^{\pi_b}_{M^*}(x, a) - Q^{\pi_b}_{\widehat{M}}(x, a)\right| \leq e_Q(x, a) V_{max}, \quad (2)$$

where $V_{max} \leq \frac{R_{max}}{1 - \gamma}$ is the maximum of the value function, and the two error functions may be derived from Hoeffding's inequality (see A.2) as

$$e_P(x, a) := \sqrt{\frac{2}{N_{\mathcal{D}}(x, a)} \log \frac{2|\mathcal{X}||\mathcal{A}|2^{|\mathcal{X}|}}{\delta}} \text{ and } e_Q(x, a) := \sqrt{\frac{2}{N_{\mathcal{D}}(x, a)} \log \frac{2|\mathcal{X}||\mathcal{A}|}{\delta}}.$$

We will also use the following definition:

Definition 1. *A policy π is said to be a policy improvement over a baseline policy π_b in an MDP $M = \langle \mathcal{X}, \mathcal{A}, P, R, \gamma \rangle$ if the following inequality holds in every state $x \in \mathcal{X}$:*

$$V^{\pi}_{M}(x) \geq V^{\pi_b}_{M}(x) \quad (3)$$

2.2 Safe Policy Improvement with Baseline Bootstrappping

Our objective is to maximize the expected return of the target policy under the constraint of improving with high probability $1 - \delta$ the baseline policy. This is known to be an NP-hard problem (Petrik et al., 2016) and some approximations are required to make it tractable. This paper builds on the Safe Policy Improvement with Baseline Bootstrapping methodology (SPIBB, Laroche et al., 2019). SPIBB finds an approximate solution to the problem by searching for a policy maximizing the expected return in the MLE

MDP $\widehat{M}$, under the constraint that the policy improvement is guaranteed in the set of plausible MDPs Ξ:

$$\underset{\pi}{\text{argmax}}\, \rho(\pi, \widehat{M}), \text{ s.t. } \forall M \in \Xi, \rho(\pi, M) \geq \rho(\pi_b, M) - \zeta \tag{4}$$

$$\Xi = \left\{ M = \langle \mathcal{X}, \mathcal{A}, R, P, \gamma \rangle \text{ s.t. } \forall x, a, \begin{array}{l} ||P(\cdot|x,a) - \widehat{P}(\cdot|x,a)||_1 \leq e_P(x,a), \\ |R(x,a) - \widehat{R}(x,a)| \leq e_P(x,a) R_{max} \end{array} \right\} \tag{5}$$

The error function e_P is such that the true MDP M^* has a high probability of at least $1-\delta$ to belong to Ξ (Iyengar, 2005; Nilim and El Ghaoui, 2005). In other terms, the objective is to optimize the target performance in $\widehat{M}$ such that its performance is ζ-approximately at least as good as π_b in the admissible MDP set, where ζ is a precision hyper-parameter. Expressed this way, the problem is still intractable. SPIBB is able to find an approximate solution within a tractable amount of time by applying a special processing to state-action pair transitions that were not sampled enough in the batch of data. The methodology consists in building a set of rare thus uncertain state-action pairs in the dataset $\mathcal{D}$, called the bootstrapped set and denoted by $\mathcal{B}$: the bootstrapped set contains all the state-action pairs $(x, a) \in \mathcal{X} \times \mathcal{A}$ whose counts in $\mathcal{D}$ are lower than a hyper-parameter $N_\wedge$. SPIBB algorithms then construct a space of allowed policies, *i.e* policies that are constrained on the bootstrapped set $\mathcal{B}$, and search for the optimal policy in this set by performing policy iteration. For example, Π_b-SPIBB is a provably-safe algorithm that assigns the baseline π_b to the state-action pairs in $\mathcal{B}$ and trains the policy on the rest. $\Pi_{\leq b}$-SPIBB is a variant that does not give more weight than π_b to the uncertain transitions.

SPIBB's principle amounts to search over a policy space constrained such that the policy improvement may be precisely assessed in M^*. Because of the hard definition of the bootstrapped set, SPIBB relies on a binary decision-making and may be too conservative. Our novel method, called Soft-SPIBB, follows the same principle, but relaxes this definition by allowing soft policy changes for the uncertain state-action pairs, and offers more flexibility than SPIBB while remaining safe.

This idea might seem similar to Conservative Policy Iteration (CPI), Trust Region Policy Optimization (TRPO), or Proximal Policy Optimization (PPO) in that it allows changes in the policy under a proximity regularization to the old policy (Kakade and Langford, 2002; Schulman et al., 2015, 2017). However, with Soft-SPIBB, the proximity constraint is tightened or relaxed according to the amount of samples supporting the policy change (see Definition 2). Additionally, CPI, TRPO, and PPO are designed for the online setting. In the batch setting we consider, they would be either too conservative if the proximity regularization is applied with respect to the fixed baseline, or would converge to the fixed point obtained when solving the MLE MDP if the proximity regularization is moving with the last policy update (Corollary 3 of Geist et al., 2019).

2.3 Linear Programming

Linear programming aims at optimizing a linear objective function under a set of linear in-equality constraints. The most common methods for solving such linear programs are

the simplex algorithm and interior point methods (IPMs, Dantzig, 1963). Even though the worst-case computational complexity of the simplex is exponential in the dimensions of the program being solved (Klee and Minty, 1972), this algorithm is efficient in practice: the number of iterations seems polynomial, and sometimes linear in the problem size (Borgwardt, 1987; Dantzig and Thapa, 2003). Nowadays, these two classes of methods continue to compete with one another: it is hard to predict the winner on a particular class of problems (Gondzio, 2012). For instance, the hyper-sparsity of the problem generally seems to favour the simplex algorithm, while IPMs can be much more efficient for large-scale linear programming.

3 Safe Policy Improvement with Soft Baseline Bootstrapping

SPIBB allows to make changes in state-action pairs where the model error does not exceed some threshold ϵ, which may be expressed as a function of $N_\wedge$. This may be seen as a hard condition on the bootstrapping mechanism: a state-action pair policy may either be changed totally, or not at all. In this paper, we propose a softer mechanism where, for a given error function, a local error budget is allocated for policy changes in each state x. Similarly to SPIBB, we search for the optimal policy in the MDP model $\widehat{M}$ estimated from the dataset $\mathcal{D}$, but we reformulate the constraint by using Definitions 2 and 3.

Definition 2. *A policy π is said to be (π_b, e, ϵ)-constrained with respect to a baseline policy π_b, an error function e, and a hyper-parameter ϵ if, for all states $x \in \mathcal{X}$, the following inequality holds:*

$$\sum_{a \in \mathcal{A}} e(x,a)\big|\pi(a|x) - \pi_b(a|x)\big| \leq \epsilon.$$

Definition 3. *A policy π is said to be π_b-advantageous in an MDP $M = \langle \mathcal{X}, \mathcal{A}, P, R, \gamma \rangle$ if the following inequality holds in every state $x \in \mathcal{X}$:*

$$\sum_{a \in \mathcal{A}} A_M^{\pi_b}(x,a)\pi(a|x) \geq 0 \tag{6}$$

Remark 1. By the policy improvement theorem, a π_b-advantageous policy is a policy improvement over π_b. The converse is not guaranteed.

3.1 Theoretical safe policy improvement bounds

We show that constraining π_b-advantageous policies appropriately allows safe policy improvements. Due to space limitation, all proofs have been moved to the appendix, Section A.

Theorem 1. *Any (π_b, e_Q, ϵ)-constrained policy π that is π_b-advantageous in $\widehat{M}$ satisfies the following inequality in every state x with probability at least $1-\delta$:*

$$V^{\pi}_{M^*}(x) - V^{\pi_b}_{M^*}(x) \geq -\frac{\epsilon V_{max}}{1-\gamma}. \tag{7}$$

Constraining the target policy to be advantageous over the baseline is a strong constraint that leads to conservative solutions. To the best of our findings, it is not possible to prove a more general bound on (π_b, e_Q, ϵ)-constrained policy improvements. However, the search over (π_b, e_P, ϵ)-constrained policies, where e_P is an error bound over the probability function P (Equation 2), allows us to guarantee safety bounds under Assumption 1, which states:

Assumption 1 *There exists a constant $\kappa < \frac{1}{\gamma}$ such that, for all state-action pairs $(x,a) \in \mathcal{X} \times \mathcal{A}$, the following inequality holds:*

$$\sum_{x',a'} e_P(x',a')\pi_b(a'|x')P^*(x'|x,a) \leq \kappa e_P(x,a). \tag{8}$$

Lemma 1, which is essential to prove Theorem 2 below, relies on Assumption 1.

Lemma 1. *Under Assumption 1, any (π_b, e_P, ϵ)-constrained policy π satisfies the following inequality for every state-action pair (x,a) with probability at least $1-\delta$:*

$$\left|Q^{\pi}_{M^*}(x,a) - Q^{\pi}_{\widehat{M}}(x,a)\right| \leq \left(\frac{e_P(x,a)}{1-\kappa\gamma} + \frac{\gamma\epsilon}{(1-\gamma)(1-\kappa\gamma)}\right) V_{max}.$$

Theorem 2. *Under Assumption 1, any (π_b, e_P, ϵ)-constrained policy π satisfies the following inequality in every state x with probability at least $1-\delta$:*

$$V^{\pi}_{M^*}(x) - V^{\pi_b}_{M^*}(x) \geq V^{\pi}_{\widehat{M}}(x) - V^{\pi_b}_{\widehat{M}}(x) - 2\left\|d^{\pi_b}_{M^*}(\cdot\,|x) - d^{\pi_b}_{\widehat{M}}(\cdot\,|x)\right\|_1 V_{max} - \frac{1+\gamma}{(1-\gamma)^2(1-\kappa\gamma)}\,\epsilon V_{max}. \tag{9}$$

Remark 2. The theorems hold for any error function e_P verifying 2 w.p. $1-\delta$.

Remark 3. Π_b-SPIBB (Laroche et al., 2019) is a particular case of Soft-SPIBB where the error function $e_P(x,a)$ equals ∞ if $(x,a) \in \mathcal{B}$ and $\frac{\epsilon}{2}$ otherwise.

Remark 4. Theorem 2 has a cubic dependency in the horizon $\frac{1}{1-\gamma}$, which is weaker than SPIBB's bounds, but allow us to safely search over more policies, when using tighter error functions. We will observe in Section 4 that Soft-SPIBB empirically outperforms SPIBB both in mean performance and in safety.

3.2 Algorithms

In this section, we design two safe policy improvement algorithms to tackle the problem defined by the Soft-SPIBB approach. They both rely on the standard policy iteration process described in Pseudo-code 1, where the policy improvement step consists in solving in every state $x \in \mathcal{X}$ the locally constrained optimization problem below:

$$\pi^{(i+1)}(\cdot|x) = \underset{\pi \in \Pi}{\operatorname{argmax}} \sum_{a \in \mathcal{A}} Q^{(i)}_{\widehat{M}}(x,a)\pi(a|x) \tag{10}$$

subject to:

Constraint 1: π being a probability: $\sum_{a\in\mathcal{A}} \pi(a|x) = 1$ and $\forall a$, $\pi(a|x) \geq 0$.
Constraint 2: π being (π_b, e, ϵ)-constrained.

Pseudo-code 1: Policy iteration process for Soft-SPIBB

Input: Baseline policy π_b, MDP model precision level ϵ and dataset $\mathcal{D}$.
Compute the model error concentration bounds $e(x,a)$.
Initialize $i = 0$ and $\pi^{(0)}(\cdot|x) = \pi_b(\cdot|x)$.
while *policy iteration stopping criterion not met* **do**
  Policy evaluation: compute $Q^{(i)}_{\widehat{M}}$ with dynamic programming.
  Policy improvement: set $\pi^{(i+1)}(\cdot|x)$ as the (exact or approximate) solution of the optimization problem defined in Equation 10.
  $i \leftarrow i + 1$
return $\pi^{(i)}$

Exact-Soft-SPIBB: The Exact-Soft-SPIBB algorithm computes the exact solution of the local optimization problem in (10) during the policy improvement step. For that, we express the problem as a Linear Program (LP) and solve it by applying the simplex algorithm. Note that we chose the simplex over IPMs as it turned out to be efficient enough for our experimental settings. For tractability in large action spaces, we reformulate the non-linear Constraint 2 as follows: we introduce $|\mathcal{A}|$ auxiliary variables $\{z(x,a)\}_{(x,a)\in\mathcal{X}\times\mathcal{A}}$, which bound from above each element of the sum. For a given $x \in \mathcal{X}$, Constraint 2 is then replaced by the following $2|\mathcal{A}| + 1$ linear constraints:

$$\forall a \in \mathcal{A}, \qquad \pi(a|x) - \pi_b(a|x) \leq z(x,a), \tag{11}$$

$$\forall a \in \mathcal{A}, \qquad -\pi(a|x) + \pi_b(a|x) \leq z(x,a), \tag{12}$$

$$\sum_a e(x,a)z(x,a) \leq \epsilon. \tag{13}$$

Approx-Soft-SPIBB: We also propose a computationally-efficient algorithm, which returns a sub-optimal target policy $\pi^{\odot}_{\sim}$. It relies on the same policy iteration, but computes an approximate solution to the optimization problem. The approach still guarantees to improve the baseline in $\widehat{M}$: $\rho(\pi^{\odot}_{\sim}, \widehat{M}) \geq \rho(\pi_b, \widehat{M})$, and falls under the Theorems

1 and 2 SPI bounds. Approx-Soft-SPIBB's local policy improvement step consists in removing, for each state x, the policy probability mass m^- from the action a^- with the lowest Q-value. Then, m^- is attributed to the action that offers the highest Q-value improvement by unit of error $\partial\epsilon$:

$$a^+ = \underset{a\in\mathcal{A}}{\operatorname{argmax}} \frac{\partial\pi(a|x)}{\partial\epsilon}\left(Q^{(i)}_{\widehat{M}}(x,a) - Q^{(i)}_{\widehat{M}}(x,a^-)\right) \tag{14}$$

$$= \underset{a\in\mathcal{A}}{\operatorname{argmax}} \frac{Q^{(i)}_{\widehat{M}}(x,a) - Q^{(i)}_{\widehat{M}}(x,a^-)}{e(x,a)} \tag{15}$$

Once m^- has been reassigned to another action with higher value, the budget is updated accordingly to the error that has been spent, and the algorithm continues with the next worst action until a stopping criteria is met: the budget is fully spent, or $a^- = a^*$, where a^* is the action with maximal state-action value. The policy improvement step of Approx-Soft-SPIBB is further formalized in Pseudo-code 2, found in the appendix, Section A.8.

Theorem 3. *The policy improvement step of Approx-Soft-SPIBB generates policies that are guaranteed to be (π_b, e, ϵ)-constrained.*

Remark 5. The argmax operator in the result returned by Pseudo-code 2 is a convergence condition. Indeed, the approximate algorithm does not guarantee that the current iteration policy search space includes the previous iteration policy, which can cause divergence: the algorithm may indefinitely cycle between two or more policies. To ensure convergence, we update $\pi^{(i)}$ with $\pi^{(i+1)}$ only if there is a local policy improvement, *i.e.* when $\mathbb{E}_{a\sim\pi^{(i+1)}(\cdot|x)}[Q^{(i)}_{\widehat{M}}(x,a)] \geq \mathbb{E}_{a\sim\pi^{(i)}(\cdot|x)}[Q^{(i)}_{\widehat{M}}(x,a)]$.

Both implementation of the Soft-SPIBB strategy comply to the requirements of Theorem 1 if only one policy iteration is performed. In Section 4.1, we empirically evaluate the 1-iteration versions, which are denoted by the '1-step' suffix.

Complexity Analysis: We study the computational complexity of Exact-Soft-SPIBB and Approx-Soft-SPIBB. The error bounds computation and the policy evaluation step are common to both algorithms, and have a complexity of $\mathcal{O}(|\mathcal{D}|)$ and $\mathcal{O}(|\mathcal{X}|^3|\mathcal{A}|^3)$ respectively. The part that differs between them is the policy improvement.

Exact-Soft-SPIBB solves the LP with the simplex algorithm, which, as recalled in Section 2.3, is in practice polynomial in the dimensions of the program being solved. In our case, the number of constraints is $3|\mathcal{A}| + 1$.

Theorem 4. *Approx-Soft-SPIBB policy improvement has a complexity of $\mathcal{O}(|\mathcal{X}||\mathcal{A}|^2)$.*

Model-free Soft-SPIBB: The Soft-SPIBB fixed point may be found in a model-free manner by fitting the Q-function to the target $y^{(i+1)}$ on the transition samples $\mathcal{D} = \langle x_j, a_j, r_j, x'_j\rangle_{j\in[\![1,N]\!]}$:

$$y^{(i+1)}_j = r_j + \gamma \sum_{a'\in\mathcal{A}} \pi^{(i+1)}(a'|x'_j)Q^{(i)}(x'_j, a'), \tag{16}$$

where $\pi^{(i+1)}$ is obtained either exactly or approximately with the policy improvement steps described in Section 3.2. Then, the policy evaluation consists in fitting $Q^{(i+1)}(x,a)$ to the set of $y_j^{(i+1)}$ values computed using the samples from $\mathcal{D}$.

Theorem 5. *Considering an MDP with exact counts, the model-based policy iteration of (Exact or Approx)-Soft-SPIBB is identical to the model-free policy iteration of (resp. Exact or Approx)-Soft-SPIBB.*

The model-free versions are less computationally efficient than their respective model-based versions, but are particularly useful since it makes function approximation easily applicable. In our infinite MDP experiment, we consider Approx-Soft-SPIBB-DQN as the DQN algorithm fitted to the model-free Approx-Soft-SPIBB targets. The Exact-Soft-SPIBB counterpart is not considered for tractability reasons. We recall that the computation of the policy improvement step relies on the estimates of an error function e_P, which may, for instance, be indirectly inferred from pseudo-counts $\widetilde{N}_{\mathcal{D}}(x,a)$ (Bellemare et al., 2016; Fox et al., 2018; Burda et al., 2019).

4 Soft-SPIBB Empirical Evaluation

This section intends to empirically validate the advances granted by Soft-SPIBB. We perform the study on two domains: on randomly generated finite MDPs, where the Soft-SPIBB algorithms are compared to several Batch RL competitors: basic RL, High Confidence Policy Improvement (Thomas, 2015, HCPI), Reward-Adjusted MDPs (Petrik et al., 2016, RaMDP), Robust MDPs (Iyengar, 2005; Nilim and El Ghaoui, 2005), and to Soft-SPIBB natural parents: Π_b-SPIBB and $\Pi_{\leq b}$-SPIBB (Laroche et al., 2019); and on a helicopter navigation task requiring function approximation, where Soft-SPIBB-DQN is compared to basic DQN, RaMDP-DQN, and SPIBB-DQN. All the benchmark algorithms had their hyper-parameters optimized beforehand. Their descriptions and the results of the hyper-parameter search is available in the appendix, Section B.2 for finite MDPs algorithms and Section C.3 for DQN-based algorithms.

In order to assess the safety of an algorithm, we run a large number of times the same experiment with a different random seed. Since the environments and the baselines are stochastic, every experiment generates a different dataset, and the algorithms are evaluated on their mean performance over the experiments, and on their conditional value at risk performance (CVaR), sometimes also called the expected shortfall: $X\%$-CVaR corresponds to the mean performance over the $X\%$ worst runs.

4.1 Random MDPs

In the random MDPs experiment, the MDP and the baseline are themselves randomly generated too. The full experimental process is formalized in Pseudo-code 3 found in the appendix, Section B.1. Because every run involves different MDP and baseline, there is the requirement for a normalized performance. This is further defined as $\overline{\rho}$:

$$\overline{\rho}(\pi, M^*) = \frac{\rho(\pi, M^*) - \rho(\pi_b, M^*)}{\rho(\pi^*, M^*) - \rho(\pi_b, M^*)}. \tag{17}$$

In order to demonstrate that Soft-SPIBB algorithms are safely improving the baselines on most MDPs in practice, we use a random generator of MDPs. All the details may be found in the appendix, Section B.1. The number of states is set to $|\mathcal{X}| = 50$, the number of actions to $|\mathcal{A}| = 4$ and the connectivity of the transition function to 4, *i.e.*, for a given state-action pair (x, a), its transition function $P(x'|x, a)$ is non-zero on four states x' only. The reward function is 0 everywhere except when entering the goal state, which is terminal and where the reward is equal to 1. The goal is chosen in such a way that the optimal value function is minimal.

Random baseline: For a randomly generated MDP M, baselines are generated according to a predefined level of performance $\eta \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$: $\rho(\pi_b, M) = \eta\rho(\pi^*, M) + (1-\eta)\rho(\tilde{\pi}, M)$, where π^* and $\tilde{\pi}$ are respectively the optimal and the uniform policies. The generation of the baseline consists in three steps: optimization, where the optimal Q-value is computed; softening, where a softmax policy is generated; and randomization, where the probability mass is randomly displaced in the baseline. The process is formally and extensively detailed in the appendix, Section B.1.

Dataset generation: Given a fixed size number of trajectories, a dataset is generated on the following modification of the original MDPs: addition of another goal state (reward is set to 1). Since the original goal state was selected so as to be the hardest to reach, the new one, which is selected uniformly, is necessarily a better goal.

Complexity empirical analysis: In Figure 1, we show an empirical confirmation of the complexity results on the gridworld task. Exact-Soft-SPIBB has a linear dependency in the number of actions. We also notice that Approx-Soft-SPIBB runs much faster: even faster than Π_b-SPIBB, and 2 times slower than basic RL. Note that the policy improvement step is by design exactly linearly dependent on the number of states $|\mathcal{X}|$. This is the reason why we do not report experiments on the dependency on $|\mathcal{X}|$. We do not report complexity empirical analysis of the other competitors because we do not pretend to have optimal implementations of them, and the purpose of this analysis is to show that Approx-Soft-SPIBB solves the tractability issues of Exact-Soft-SPIBB. In Theory, Robust MDPs and HCPI are more complex by at least an order of magnitude.

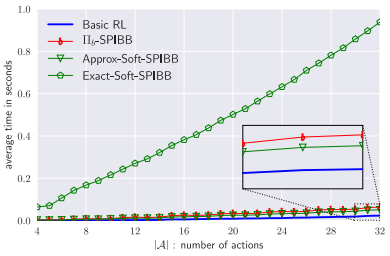


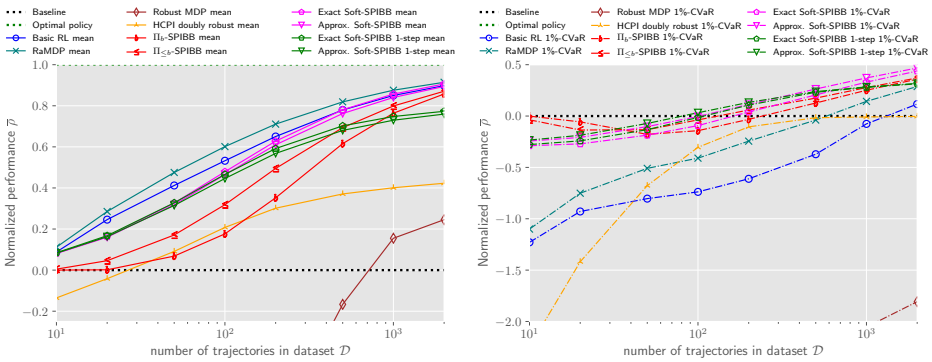
Fig. 1. Average time to convergence.

Benchmark results: Figures 2a and 2b respectively report the mean and 1%-CVAR performances with a strong baseline ($\eta = 0.9$). Robust MDPs and HCPI perform poorly and are not further discussed. Basic RL and RaMDP win the benchmark in mean, but fail to do it safely, contrary to Soft-SPIBB and SPIBB algorithms. Exact-Soft-SPIBB is slightly better than Approx-Soft-SPIBB in mean, but also slightly worse in safety. Still in comparison to Approx-Soft-SPIBB, Exact-Soft-SPIBB's performance

does not justify the computational complexity increase and will not be further discussed. Approx-Soft-SPIBB demonstrates a significant improvement over SPIBB methods, both in mean and in safety. Finally, the comparison of Approx-Soft-SPIBB with Approx-Soft-SPIBB 1-step shows that the safety is not improved in practice, and that the asymptotic optimality is compromised when the dataset becomes larger.

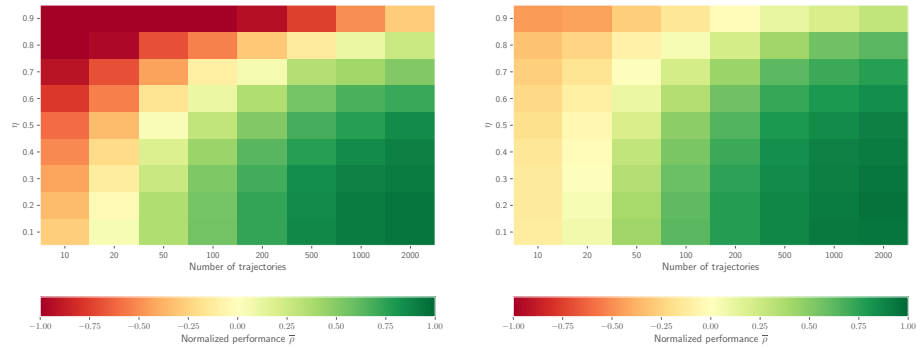
Sensitivity to the baseline: We continue the analysis with a heatmap representation as a function of the strength of the baseline: Figures 3a and 3b display heatmaps of the 0.1%-CVaR performance for RaMDP and Approx-Soft-SPIBB ($\epsilon = 2$) respectively. The colour of a cell indicates the improvement over the baseline normalized with respect to the optimal performance: red, yellow, and green respectively mean below, equal to, and above baseline performance. We observe that RaMDP is unsafe for strong baselines (high η values) and small datasets, while Soft-SPIBB methods become slightly unsafe only with $\eta = 0.9$ and less than 20 trajectories, but are safe everywhere else.

Sensitivity to hyper-parameters: We carry on with 1%-CVaR performance heatmaps as a function of the hyper-parameters for RaMDP (Figure 4a) and Approx-Soft-SPIBB (Figure 4b) in the hardest scenario ($\eta = 0.9$). The choice of 1%-CVaR instead of 0.1%-CVaR is justified by the fact that the 0.1%-CVaR RaMDP heatmap is almost completely red, which would not allow us to notice the interesting thresholding behaviour: when $\kappa_{adj} \geq 0.0035$, RaMDP becomes over-conservative to the point of not trying to reach the goal anymore. In contrast, Approx-Soft-SPIBB behaves more smoothly with respect to its hyper-parameter, its optimal value being in interval [0.5,2], depending on the safety/performance trade-off one wants to achieve. In the appendix, Section B.3, the interested reader may find the corresponding heatmaps for all Soft-SPIBB algorithms for mean and 1%-CVaR performances. In particular, we may observe that, despite not having as strong theoretical guarantees as their 1-step versions, the Soft-SPIBB algorithms demonstrate similar CVaR performances.



(a) Mean: $\eta = 0.9$, $\epsilon = 2$ (b) 1%-CVAR: $\eta = 0.9$, $\epsilon = 2$

Fig. 2. Benchmark on Random MDPs domain: mean and 1%-CVAR performances for a hard scenario ($\eta = 0.9$) and Soft-SPIBB with $\epsilon = 2$

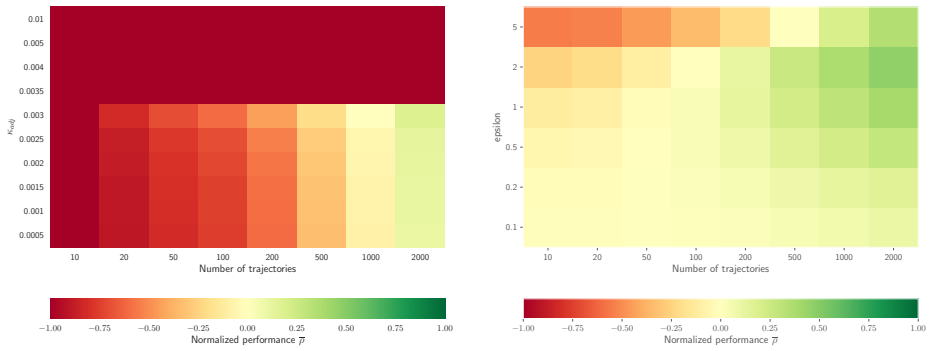


(a) 0.1%-CVaR: RaMDP (b) 0.1%-CVaR: Approx-Soft-SPIBB, $\epsilon = 2$

Fig. 3. Influence of η on Random MDPs domain: 0.1%-CVaR heatmaps as a function of η

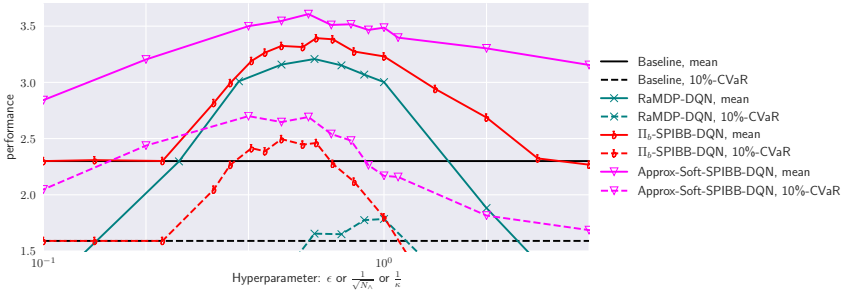
4.2 Helicopter domain

To assess our algorithms on tasks with more complex state spaces, making the use of function approximation inevitable, we apply them to a helicopter navigation task (Figure 5(c)). The helicopter's start point is randomly picked in the teal region, its initial velocity is random as well. The agent can choose to apply or not a fixed amount of thrust forward and backward in the two dimensions, resulting in 9 actions total. An episode ends when the agent reaches the boundary of the blue box or has a speed larger than some maximal value. In the first case, it receives a reward based on its position with respect to the top right corner of the box (the exact reward value is chromatically indicated in the figure). In the second, it gets a reward of -1. The dynamics of the helicopter obey Newton's second law with an additive centered Gaussian noise applied

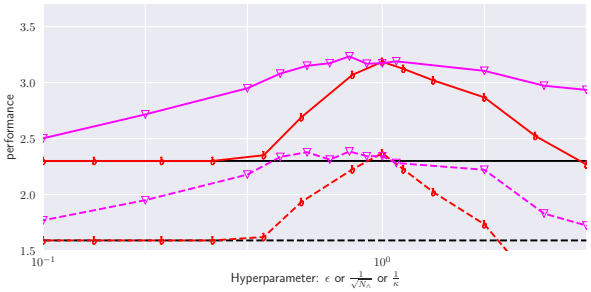


(a) 1%-CVaR: RaMDP, $\eta = 0.9$ (b) 1%-CVaR: Approx-Soft-SPIBB, $\eta = 0.9$

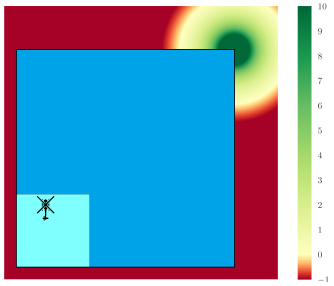
Fig. 4. Sensitivity to hyperparameter on Random MDPs: 1%-CVaR heatmaps for $\eta = 0.9$



(a) Helicopter benchmark with $|\mathcal{D}| = 10,000$



(b) Helicopter benchmark with $|\mathcal{D}| = 3,000$



(c) Helicopter domain

Fig. 5. Helicopter: mean and 10%-CVaR as a function of the hyper-parameter value

to its position and velocity. We refer the reader to the appendix, Section C.1 for the detailed specifications. We generated a baseline by training online a DQN (Mnih et al., 2015) and applying a softmax on the learnt Q-network. During training, a discount factor of 0.9 is used, but the reported results show the undiscounted return obtained by the agent.

The experiments consist in 300 training runs (necessary to obtain reasonable estimates of algorithms' safety, the full training procedure is described in the appendix, Section C.3) of RaMDP-DQN, SPIBB-DQN and Approx-Soft-SPIBB-DQN, for different values of their hyper-parameters (resp. κ, $N_\wedge$ and ϵ). We note that for $\kappa = 0$, $N_\wedge = 0$ or $\epsilon = +\infty$, those three algorithms become standard DQN, and that for $N_\wedge = \infty$ or $\epsilon = 0$, the SPIBB and Soft-SPIBB algorithms produce a copy of the baseline. The three algorithms rely on some estimates of the state-action counts. In this work, we used a pseudo-count estimate heuristic based on Euclidean distance, also detailed in Section C.3. For scalability, we may consider several pseudo-count methodologies from the literature Bellemare et al. (2016); Fox et al. (2018). This is left for future work.

The results of our evaluation can be found in Figure 5, where we plot the mean and 10%-CVaR performances of the different algorithms for two sizes of datasets (more results may be found in the appendix, Section C.4). In order to provide meaningful

comparisons, the abscissa represents the different hyper-parameters transformed to account for their dimensional homogeneity (except for a scaling factor). Both Approx-Soft-SPIBB-DQN and SPIBB-DQN outperform RaMDP-DQN by a large margin on the datasets of size 10,000. On the smaller datasets, RaMDP-DQN performs very poorly and does not even appear on the graph. For the same reason, vanilla DQN (mean: 0.22 and 10%-CVaR: -1 with $|\mathcal{D}| = 10,000$) does not appear on any of the graphs. The two SPIBB algorithms significantly outperform the baseline both in mean and 10%-CVaR. At their best hyper-parameter value, their 10%-CVaR is actually better than the mean performance of the baseline. Approx-Soft-SPIBB-DQN performs better than SPIBB-DQN both in mean and 10%-CVaR performances. Finally, it is less sensitive than SPIBB-DQN with respect to their respective hyperparameters, and demonstrates a better stability over different dataset sizes. That stability is a useful property as it reduces the requirement for hyper-parameter optimization, which is crucial for Batch RL.

5 Conclusion

We study the problem of safe policy improvement in a Batch RL setting. Building on the SPIBB methodology, we relax the constraints of the policy search to propose a family of algorithms coined Soft-SPIBB. We provide proofs of safety and of computational efficiency for an algorithm called Approx-Soft-SPIBB based on the search of an approximate solution that does not compromise the safety guarantees. We support the theoretical work with an extensive empirical analysis where Approx-Soft-SPIBB shines as the best compromise average performance vs. safety. We further develop Soft-SPIBB in a model-free manner which helps its application to function approximation. Despite the lack of theoretical safety guarantees with function approximation, we observe in our experiments where the function approximation is modelled as a neural network, that Soft-SPIBB allows safe policy improvement in practice and significantly outperforms the competing algorithms both in safety and in performance.

Bibliography

Bellemare, M., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., and Munos, R. (2016). Unifying count-based exploration and intrinsic motivation. In *Proceedings of the 29th Advances in Neural Information Processing Systems (NIPS)*.

Borgwardt, K. H. (1987). *The Simplex Method: A Probabilistic Analysis*. Springer-Verlag Berlin Heidelberg.

Burda, Y., Edwards, H., Storkey, A., and Klimov, O. (2019). Exploration by random network distillation. In *Proceedings of the 7th International Conference on Learning Representations (ICLR)*.

Dantzig, G. (1963). *Linear programming and extensions*. Rand Corporation Research Study. Princeton Univ. Press, Princeton, NJ.

Dantzig, G. B. and Thapa, M. N. (2003). *Linear Programming 2: Theory and Extensions*. Springer-Verlag New York.

Ernst, D., Geurts, P., and Wehenkel, L. (2005). Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6(Apr):503–556.

Fox, L., Choshen, L., and Loewenstein, Y. (2018). Dora the explorer: Directed outreaching reinforcement action-selection. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)*.

Geist, M., Scherrer, B., and Pietquin, O. (2019). A theory of regularized markov decision processes. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*.

Gondzio, J. (2012). Interior point methods 25 years later. *European Journal of Operational Research*.

Guez, A., Vincent, R. D., Avoli, M., and Pineau, J. (2008). Adaptive treatment of epilepsy via batch-mode reinforcement learning. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, pages 1671–1678.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *arXiv preprint arXiv:1502.01852*.

Iyengar, G. N. (2005). Robust dynamic programming. *Mathematics of Operations Research*.

Kakade, S. and Langford, J. (2002). Approximately optimal approximate reinforcement learning. In *Proceedings of the 19th International Conference on Machine Learning (ICML)*, volume 2, pages 267–274.

Klee, V. and Minty, G. J. (1972). How good is the simplex algorithm? In Shisha, O., editor, *Inequalities*, volume III, pages 159–175. Academic Press, New York.

Lange, S., Gabel, T., and Riedmiller, M. (2012). *Batch Reinforcement Learning*, pages 45–73. Springer Berlin Heidelberg, Berlin, Heidelberg.

Laroche, R., Trichelair, P., and Tachet des Combes, R. (2019). Safe policy improvement with baseline bootstrapping. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529.

Nilim, A. and El Ghaoui, L. (2005). Robust control of markov decision processes with uncertain transition matrices. *Operations Research*.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in pytorch. In *NIPS-W*.

Petrik, M., Ghavamzadeh, M., and Chow, Y. (2016). Safe policy improvement by minimizing robust baseline regret. In *Proceedings of the 29th Advances in Neural Information Processing Systems (NIPS)*.

Riedmiller, M. (2005). Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method. In *Proceedings of the 16th European Conference on Machine Learning (ECML)*, pages 317–328. Springer.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. (2015). Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Simão, T. D. and Spaan, M. T. J. (2019). Safe policy improvement with baseline bootstrapping in factored environments. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*.

Singh, S. P., Kearns, M. J., Litman, D. J., and Walker, M. A. (1999). Reinforcement learning for spoken dialogue systems. In *Proceedings of the 13st Advances in Neural Information Processing Systems (NIPS)*, pages 956–962.

Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. The MIT Press.

Thomas, P. S. (2015). *Safe reinforcement learning*. PhD thesis, Stanford university.

Tieleman, T. and Hinton, G. (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31.

van Hasselt, H., Guez, A., and Silver, D. (2015). Deep reinforcement learning with double q-learning. *CoRR*, abs/1509.06461.

Weissman, T., Ordentlich, E., Seroussi, G., Verdu, S., and Weinberger, M. J. (2003). Inequalities for the l1 deviation of the empirical distribution. *Hewlett-Packard Labs, Tech. Rep*.

A Proofs

A.1 Preliminaries

We start by recalling the various definitions used in the proofs.

Definition 1. *A policy π is said to be a policy improvement over a baseline policy π_b in an MDP $M = \langle \mathcal{X}, \mathcal{A}, P, R, \gamma \rangle$ if the following inequality holds in every state $x \in \mathcal{X}$:*

$$V_M^{\pi}(x) \geq V_M^{\pi_b}(x) \tag{18}$$

Definition 2. *A policy π is said to be π_b-advantageous in an MDP $M = \langle \mathcal{X}, \mathcal{A}, P, R, \gamma \rangle$ if the following inequality holds in every state $x \in \mathcal{X}$:*

$$\sum_{a \in \mathcal{A}} A_M^{\pi_b}(x, a)\pi(a|x) \geq 0 \tag{19}$$

where $A_M^{\pi_b}(x, a) = Q_M^{\pi_b}(x, a) - V_M^{\pi_b}(x)$ quantifies the advantage (or disadvantage) of taking action a in state x instead of following policy π_b.

Definition 3. *A policy π is said to be (π_b, e, ϵ)-constrained for baseline policy π_b, error function e, and a hyper-parameter ϵ if, for all states $x \in \mathcal{X}$, the following inequality holds:*

$$\sum_{a \in \mathcal{A}} e(x, a)\big|\pi(a|x) - \pi_b(a|x)\big| \leq \epsilon.$$

A.2 Error Bounds

The difference between an estimated parameter and the true one can be bounded using concentration bounds (or equivalently, Hoeffding's inequality) applied to the state-action counts in $\mathcal{D}$ (Petrik et al., 2016; Laroche et al., 2019). Specifically, the following inequalities hold with probability at least $1 - \delta$ for any state-action pair $(x, a) \in \mathcal{X} \times \mathcal{A}$:

$$||P^*(\cdot|x, a) - \widehat{P}(\cdot|x, a)||_1 \leq e_P(x, a) \tag{20}$$

$$|R^*(x, a) - \widehat{R}(x, a)| \leq e_P(x, a)R_{max} \tag{21}$$

$$\big|Q_{M^*}^{\pi_b}(x, a) - Q_{\widehat{M}}^{\pi_b}(x, a)\big| \leq e_Q(x, a)V_{max} \tag{22}$$

where:

$$e_P(x, a) := \sqrt{\frac{2}{N_{\mathcal{D}}(x, a)} \log \frac{2|\mathcal{X}||\mathcal{A}|2^{|\mathcal{X}|}}{\delta}} \tag{23}$$

$$e_Q(x, a) := \sqrt{\frac{2}{N_{\mathcal{D}}(x, a)} \log \frac{2|\mathcal{X}||\mathcal{A}|}{\delta}}. \tag{24}$$

Proof. The three inequalities can be proved similarly to (Petrik et al., 2016, Proposition 9). We only detail the proof for (22): for any $(x,a) \in \mathcal{X} \times \mathcal{A}$, and from the two-sided Hoeffding's inequality,

$$\begin{aligned}
&\mathbb{P}\left(\left|Q^{\pi_b}_{M^*}(x,a) - Q^{\pi_b}_{\widehat{M}}(x,a)\right| > e_Q(x,a)V_{max}\right) \\
&= \mathbb{P}\left(\frac{\left|Q^{\pi_b}_{M^*}(x,a) - Q^{\pi_b}_{\widehat{M}}(x,a)\right|}{2V_{max}} > \sqrt{\frac{1}{2N_{\mathcal{D}}(x,a)}\log\frac{2|\mathcal{X}||\mathcal{A}|}{\delta}}\right) \\
&\leq 2\exp\left(-2N_{\mathcal{D}}(x,a)\frac{1}{2N_{\mathcal{D}}(x,a)}\log\frac{2|\mathcal{X}||\mathcal{A}|}{\delta}\right) \\
&\leq \frac{\delta}{|\mathcal{X}||\mathcal{A}|}
\end{aligned}$$

By summing all $|\mathcal{X}||\mathcal{A}|$ state-action pairs error probabilities lower than $\frac{\delta}{|\mathcal{X}||\mathcal{A}|}$, we obtain (22).

A.3 Proof of Theorem 1

In this section, we prove Theorem 1:

Theorem 1. *Any (π_b, e_Q, ϵ)-constrained policy π that is π_b-advantageous in $\widehat{M}$ satisfies the following inequality in every state x with probability at least $1-\delta$:*

$$V^{\pi}_{M^*}(x) - V^{\pi_b}_{M^*}(x) \geq -\frac{\epsilon V_{max}}{1-\gamma}. \tag{25}$$

Proof. We start from Proposition 1 with $M = M^*$, $\pi_1 = \pi$, and $\pi_2 = \pi_b$:

$$\begin{aligned}
V^{\pi}_{M^*} - V^{\pi_b}_{M^*} &= Q^{\pi_b}_{M^*}(\pi - \pi_b)\, d^{\pi}_{M^*} && (26)\\
&= \left(Q^{\pi_b}_{M^*} - Q^{\pi_b}_{\widehat{M}} + Q^{\pi_b}_{\widehat{M}}\right)(\pi - \pi_b)\, d^{\pi}_{M^*} && (27)\\
&= \left(Q^{\pi_b}_{M^*} - Q^{\pi_b}_{\widehat{M}}\right)(\pi - \pi_b)\, d^{\pi}_{M^*} + Q^{\pi_b}_{\widehat{M}}(\pi - \pi_b)\, d^{\pi}_{M^*}. && (28)
\end{aligned}$$

The first term is bounded by $\frac{\epsilon V_{max}}{1-\gamma}$ thanks to inequality 24, the (π_b, e_Q, ϵ)-constrained policy property and Holder's inequality:

$$\begin{aligned}
&\left\| \left(Q^{\pi_b}_{M^*} - Q^{\pi_b}_{\widehat{M}}\right)(\pi - \pi_b)\, d^{\pi}_{M^*} \right\|_{\infty} \leq \left\| \left(Q^{\pi_b}_{M^*} - Q^{\pi_b}_{\widehat{M}}\right)(\pi - \pi_b) \right\|_{\infty} \left\| d^{\pi}_{M^*} \right\|_1 && (29)\\
&\leq \max_x \sum_a \left(Q^{\pi_b}_{M^*}(x,a) - Q^{\pi_b}_{\widehat{M}}(x,a)\right)\left(\pi(a|x) - \pi_b(a|x)\right)\frac{1}{1-\gamma} && (30)\\
&\leq \frac{\epsilon V_{max}}{1-\gamma}. && (31)
\end{aligned}$$

Let us now prove that the second term of Equation 28 is positive. It is the product of vector $Q^{\pi_b}_{\widehat{M}}(\pi - \pi_b)$ with matrix $d^{\pi}_{M^*}$. All the terms of $d^{\pi}_{M^*}$ are positive so it suffices

to show that each element of the vector is positive. We have for each $x \in \mathcal{X}$:

$$\left(Q^{\pi_b}_{\widehat{M}}(\pi - \pi_b)\right)(x) = \sum_a Q^{\pi_b}_{\widehat{M}}(x,a)\left(\pi(a|x) - \pi_b(a|x)\right) \tag{32}$$

$$= \sum_a Q^{\pi_b}_{\widehat{M}}(x,a)\pi(a|x) - V^{\pi_b}_{\widehat{M}}(x) \tag{33}$$

$$= \sum_{a\in\mathcal{A}} A^{\pi_b}_M(x,a)\pi(a|x) \geq 0 \tag{34}$$

where the last inequality comes from π being π_b-advantageous in $\widehat{M}$. This concludes the proof. □

A.4 Proof of Theorem 2

In this section, we prove Theorem 2 from the main text. Let us start with:

Proposition 1. *Let $M = \langle \mathcal{X}, \mathcal{A}, P, R, \gamma\rangle$ be an MDP, and π_1 and π_2 be two policies defined on this MDP. We have:*

$$V^{\pi_1}_M - V^{\pi_2}_M = Q^{\pi_1}_M(\pi_1 - \pi_2)\, d^{\pi_2}_M = Q^{\pi_2}_M(\pi_1 - \pi_2)\, d^{\pi_1}_M, \tag{35}$$

where d^π_M is a matrix that assigns to element $d^\pi_M(x'|x)$ the expectation of the discounted sum of visits to state x', starting from x, and when following policy π in MDP M (it is actually only dependent on the transition dynamics P). Formally, it is written as:

$$d^\pi_M(x'|x) = \sum_{t=0}^{\infty} \gamma^t \mathbb{P}(X_t = x' | X_t \sim P\pi X_{t-1}, X_0 = x) \tag{36}$$

Proof.

$$V^{\pi_1}_M - V^{\pi_2}_M = (R + \gamma V^{\pi_1}_M P)\pi_1 - (R + \gamma V^{\pi_2}_M P)\pi_2 \tag{37}$$

$$= (R + \gamma V^{\pi_1}_M P)(\pi_1 - \pi_2) + \gamma(V^{\pi_1}_M - V^{\pi_2}_M)P\pi_2 \tag{38}$$

$$= Q^{\pi_1}_M(\pi_1 - \pi_2) + \gamma(V^{\pi_1}_M - V^{\pi_2}_M)P\pi_2 \tag{39}$$

$$(V^{\pi_1}_M - V^{\pi_2}_M)(\mathbb{I} - \gamma P\pi_2) = Q^{\pi_1}_M(\pi_1 - \pi_2) \tag{40}$$

$$V^{\pi_1}_M - V^{\pi_2}_M = Q^{\pi_1}_M(\pi_1 - \pi_2)(\mathbb{I} - \gamma P\pi_2)^{-1} \tag{41}$$

The last line is obtained because $P\pi_2$ is a stochastic matrix and $0 \leq \gamma < 1$. As a result, $\mathbb{I} - \gamma P\pi_2$ is invertible and is indeed the state distribution matrix $d^{\pi_2}_M$. The other equality: $V^{\pi_1}_M - V^{\pi_2}_M = Q^{\pi_2}_M(\pi_1 - \pi_2)d^{\pi_1}_M$ is obtained in a symmetrical way. □

Proposition 2. *Let M_1 and M_2 be two MDPs: $M_1 = \langle \mathcal{X}, \mathcal{A}, P_1, R_1, \gamma\rangle$ and $M_2 = \langle \mathcal{X}, \mathcal{A}, P_2, R_2, \gamma\rangle$. Let π_1, π_2 be two policies defined on these MDPs, then:*

$$V^{\pi_1}_{M_1} - V^{\pi_2}_{M_1} = V^{\pi_1}_{M_2} - V^{\pi_2}_{M_2} + Q^{\pi}_{M_2}(\pi_1 - \pi_2)\left(d^{\pi_2}_{M_1} - d^{\pi_2}_{M_2}\right) \tag{42}$$

$$+ \left(Q^{\pi}_{M_1} - Q^{\pi}_{M_2}\right)(\pi_1 - \pi_2)\, d^{\pi_2}_{M_1} \tag{43}$$

Proof. From Proposition 1:

$$
\begin{aligned}
V^{\pi_1}_{M_1} - V^{\pi_2}_{M_1} &= Q^{\pi_1}_{M_1} (\pi_1 - \pi_2) d^{\pi_2}_{M_1} && (44)\\
&= \left(Q^{\pi_1}_{M_2} + Q^{\pi_1}_{M_1} - Q^{\pi_1}_{M_2}\right) (\pi_1 - \pi_2) d^{\pi_2}_{M_1} && (45)\\
&= Q^{\pi_1}_{M_2} (\pi_1 - \pi_2) d^{\pi_2}_{M_1} + \left(Q^{\pi_1}_{M_1} - Q^{\pi_1}_{M_2}\right) (\pi_1 - \pi_2) d^{\pi_2}_{M_1} && (46)\\
&= Q^{\pi_1}_{M_2} (\pi_1 - \pi_2) \left(d^{\pi_2}_{M_2} + d^{\pi_2}_{M_1} - d^{\pi_2}_{M_2}\right) + \left(Q^{\pi_1}_{M_1} - Q^{\pi_1}_{M_2}\right) (\pi_1 - \pi_2) d^{\pi_2}_{M_1} && (47)\\
&= Q^{\pi_1}_{M_2} (\pi_1 - \pi_2) d^{\pi_2}_{M_2} + Q^{\pi_1}_{M_2} (\pi_1 - \pi_2) \left(d^{\pi_2}_{M_1} - d^{\pi_2}_{M_2}\right) && (48)\\
&\qquad + \left(Q^{\pi_1}_{M_1} - Q^{\pi_1}_{M_2}\right) (\pi_1 - \pi_2) d^{\pi_2}_{M_1} && (49)
\end{aligned}
$$

The first term of Equation 49 is transformed by applying Proposition 1 to M_2:

$$
Q^{\pi_1}_{M_2} (\pi_1 - \pi_2) d^{\pi_2}_{M_2} = V^{\pi_1}_{M_2} - V^{\pi_2}_{M_2}, \tag{50}
$$

which concludes the proof. □

The proof of the theorem requires the following assumption (see the main text for a discussion on it):

Assumption 1 *There exists a constant $\kappa < \frac{1}{\gamma}$ such that, for all state-action pairs $(x, a) \in \mathcal{X} \times \mathcal{A}$ the following inequality holds:*

$$
\sum_{x',a'} e_P(x', a') \pi_b(a'|x') P^*(x'|x, a) \leq \kappa e(x, a) \tag{51}
$$

That assumption allows to prove Lemma 1:

Lemma 1. *Under Assumption 1, any (π_b, e_P, ϵ)-constrained policy π satisfies the following inequality for every state-action pair (x, a) with probability at least $1 - \delta$:*

$$
\left|Q^{\pi}_{M^*}(x, a) - Q^{\pi}_{\widehat{M}}(x, a)\right| \leq \left(\frac{e_P(x, a)}{1 - \kappa\gamma} + \frac{\gamma\epsilon}{(1 - \gamma)(1 - \kappa\gamma)}\right) V_{max}.
$$

Proof sketch. Let $Q^*_0 = 0$, $\widehat{Q}_0 = 0$, $Q^*_{t+1} = \mathcal{B}^*(Q^*_t)$ and $\widehat{Q}_{t+1} = \widehat{\mathcal{B}}(\widehat{Q}_t)$, where $\mathcal{B}^*$ and $\widehat{\mathcal{B}}$ denote the Bellman operators for policy π in the true and estimated MDP. We proceed by induction to prove inequality 52 on $|Q^*_t(x, a) - \widehat{Q}_t(x, a)|$:

$$
\begin{aligned}
Q^*_{t+1}(x, a) - \widehat{Q}_{t+1}(x, a) &= R^*(x, a) - \widehat{R}(x, a) && (52)\\
&\quad + \sum_{x',a'} \gamma \left(Q^*_t(x', a') - \widehat{Q}_t(x', a')\right) \pi(a'|x') P^*(x'|x, a)\\
&\quad + \sum_{x',a'} \gamma \widehat{Q}_t(x', a') \pi(a'|x') \left(P^*(x'|x, a) - \widehat{P}(x'|x, a)\right).
\end{aligned}
$$

By applying the inductive hypothesis and inequality 1 (which holds true with probability at least $1-\delta$):

$$\begin{aligned}\left|Q^*_{t+1}(x,a) - \widehat{Q}_{t+1}(x,a)\right| \leq e_P(x,a)V_{max} + \frac{\gamma^2\epsilon V_{max}}{(1-\gamma)(1-\kappa\gamma)} \\ + \sum_{x',a'} \frac{\gamma e_P(x',a')}{1-\kappa\gamma}\pi_b(a'|x')P^*(x'|x,a)V_{max} \qquad (53) \\ + \sum_{x',a'} \frac{\gamma e_P(x',a')}{1-\kappa\gamma}|\pi(a'|x') - \pi_b(a'|x')|P^*(x'|x,a)V_{max}.\end{aligned}$$

The (π_b, e_P, ϵ)-constrained property of π ensures that

$$\forall x' \in \mathcal{X},\ \sum_{a'} e_P(x',a')|\pi(a'|x') - \pi_b(a'|x')| \leq \epsilon$$

Applying Assumption 1 and some algebraic manipulations allows to conclude the induction. Taking the limit proves the lemma since $(Q^*_t, \widehat{Q}_t) \to (Q^\pi_{M^*}, Q^\pi_{\widehat{M}})$. □

Proof. We let $\mathcal{B}^*$ and $\widehat{\mathcal{B}}$ denote the Bellman operators for policy π in the true and estimated MDP. We have *e.g.*:

$$\mathcal{B}^*(Q)(x,a) = R^*(x,a) + \sum_{x',a'} \gamma Q(x',a')\pi(a'|x')P^*(x'|x,a).$$

We now proceed by induction. Let $Q^*_0 = 0$, $\widehat{Q}_0 = 0$, $Q^*_{t+1} = \mathcal{B}^*(Q^*_t)$ and $\widehat{Q}_{t+1} = \widehat{\mathcal{B}}(\widehat{Q}_t)$. We wish to prove that :

$$|Q^*_t(x,a) - \widehat{Q}_t(x,a)| \leq \left(\frac{e_P(x,a)}{1-\kappa\gamma} + \frac{\gamma\epsilon}{(1-\gamma)(1-\kappa\gamma)}\right) V_{max}.$$

By definition we have:

$$\begin{aligned}Q^*_{t+1}(x,a) - \widehat{Q}_{t+1}(x,a) &= \mathcal{B}^*(Q^*_t)(x,a) - \widehat{\mathcal{B}}(\widehat{Q}_t)(x,a) \qquad (54) \\ &= R^*(x,a) - \widehat{R}(x,a) + \sum_{x',a'} \gamma Q^*_t(x',a')\pi(a'|x')P^*(x'|x,a) \\ &\quad - \sum_{x',a'} \gamma \widehat{Q}_t(x',a')\pi(a'|x')\widehat{P}(x'|x,a) \qquad (55) \\ &= R^*(x,a) - \widehat{R}(x,a) \\ &\quad + \sum_{x',a'} \gamma \left(Q^*_t(x',a') - \widehat{Q}_t(x',a')\right)\pi(a'|x')P^*(x'|x,a) \\ &\quad + \sum_{x',a'} \gamma \widehat{Q}_t(x',a')\pi(a'|x')\left(P^*(x'|x,a) - \widehat{P}(x'|x,a)\right) \\ &\qquad (56)\end{aligned}$$

We apply inequality 21 to the first term, the inductive hypothesis at time t to the second and inequality 20 and Holder's inequality to the third to get:

$$
\begin{aligned}
&|Q^*_{t+1}(x,a) - \widehat{Q}_{t+1}(x,a)| \\
&\quad \leq e_P(x,a)R_{max} + \gamma e_P(x,a)V_{max} \\
&\qquad + \sum_{x',a'} \gamma \left(\frac{e_P(x',a')}{1-\kappa\gamma} + \frac{\gamma\epsilon}{(1-\gamma)(1-\kappa\gamma)} \right) \pi(a'|x') P^*(x'|x,a) V_{max} \qquad (57) \\
&\quad = e_P(x,a)V_{max} + \frac{\gamma^2 \epsilon V_{max}}{(1-\gamma)(1-\kappa\gamma)} \\
&\qquad + \sum_{x',a'} \frac{\gamma e_P(x',a')}{1-\kappa\gamma} \pi(a'|x') P^*(x'|x,a) V_{max} \qquad (58) \\
&\quad = e_P(x,a)V_{max} + \frac{\gamma^2 \epsilon V_{max}}{(1-\gamma)(1-\kappa\gamma)} \\
&\qquad + \sum_{x',a'} \frac{\gamma e_P(x',a')}{1-\kappa\gamma} \left(\pi(a'|x') - \pi_b(a'|x') + \pi_b(a'|x')\right) P^*(x'|x,a) V_{max} \qquad (59) \\
&\quad \leq e_P(x,a)V_{max} + \frac{\gamma^2 \epsilon V_{max}}{(1-\gamma)(1-\kappa\gamma)} \\
&\qquad + \sum_{x',a'} \frac{\gamma e_P(x',a')}{1-\kappa\gamma} \pi_b(a'|x') P^*(x'|x,a) V_{max} \\
&\qquad + \sum_{x',a'} \frac{\gamma e_P(x',a')}{1-\kappa\gamma} |\pi(a'|x') - \pi_b(a'|x')| P^*(x'|x,a) V_{max} \qquad (60)
\end{aligned}
$$

By the (π_b, e_P, ϵ)-constrained property, we know that $\forall x', \sum_{a'} e_P(x',a')|\pi(a'|x') - \pi_b(a'|x')| \leq \epsilon$. Applying Holder's inequality and combining the second and fourth term, we find:

$$
\begin{aligned}
|Q^*_{t+1}(x,a) - \widehat{Q}_{t+1}(x,a)| \leq e_P(x,a)V_{max} + \frac{\gamma\epsilon V_{max}}{(1-\gamma)(1-\kappa\gamma)} \\
+ \sum_{x',a'} \frac{\gamma e_P(x',a')}{1-\kappa\gamma} \pi_b(a'|x') P^*(x'|x,a) V_{max} \qquad (61)
\end{aligned}
$$

By Assumption 1, we know that $\forall x, \sum_{x',a'} e_P(x',a')\pi_b(a'|x')P^*(x'|x,a) \leq \kappa e_P(x,a)$. This gives:

$$|Q^*_{t+1}(x,a) - \widehat{Q}_{t+1}(x,a)| \leq e_P(x,a)V_{max} + \frac{\gamma\epsilon V_{max}}{(1-\gamma)(1-\kappa\gamma)} + \frac{\kappa\gamma e_P(x,a)}{1-\kappa\gamma}V_{max} \tag{62}$$

$$= \frac{e_P(x,a)V_{max}}{1-\kappa\gamma} + \frac{\gamma\epsilon V_{max}}{(1-\gamma)(1-\kappa\gamma)} \tag{63}$$

$$= \left(\frac{e_P(x,a)}{1-\kappa\gamma} + \frac{\gamma\epsilon}{(1-\gamma)(1-\kappa\gamma)}\right) V_{max}, \tag{64}$$

which concludes the induction. From standard properties of the Bellman operator, taking the limit proves the lemma. □

Theorem 2. *Under Assumption 1, any (π_b, e_P, ϵ)-constrained policy π satisfies the following inequality in every state x with probability at least $1-\delta$:*

$$V^{\pi}_{M^*}(x) - V^{\pi_b}_{M^*}(x) \geq V^{\pi}_{\widehat{M}}(x) - V^{\pi_b}_{\widehat{M}}(x) - 2\left\|d^{\pi_b}_{M^*}(\cdot\,|x) - d^{\pi_b}_{\widehat{M}}(\cdot\,|x)\right\|_1 V_{max} - \frac{1+\gamma}{(1-\gamma)^2(1-\kappa\gamma)}\,\epsilon V_{max}. \tag{65}$$

Proof sketch. We first prove the following:

$$V^{\pi}_{M^*} - V^{\pi_b}_{M^*} = V^{\pi}_{\widehat{M}} - V^{\pi_b}_{\widehat{M}} + Q^{\pi}_{\widehat{M}}(\pi - \pi_b)\left(d^{\pi_b}_{M^*} - d^{\pi_b}_{\widehat{M}}\right) + \left(Q^{\pi}_{M^*} - Q^{\pi}_{\widehat{M}}\right)(\pi - \pi_b)\, d^{\pi_b}_{M^*}. \tag{66}$$

We then apply Holder's inequality twice to the second term:

$$\left\|Q^{\pi}_{\widehat{M}}(\pi - \pi_b)\left(d^{\pi_b}_{M^*} - d^{\pi_b}_{\widehat{M}}\right)(\cdot\,|x)\right\|_1 \leq 2\left\|\left(d^{\pi_b}_{M^*} - d^{\pi_b}_{\widehat{M}}\right)(\cdot\,|x)\right\|_1 V_{max}. \tag{67}$$

Afterwards, thanks to Assumption 1, we may use Lemma 1 to bound the third term in function of $V_{max}, \epsilon, \gamma$, and κ:

$$\left\|\left(Q^{\pi}_{M^*} - Q^{\pi}_{\widehat{M}}\right)(\pi - \pi_b)\, d^{\pi_b}_{M^*}(\cdot\,|x)\right\|_1 \leq \frac{1+\gamma}{(1-\gamma)^2(1-\kappa\gamma)}\,\epsilon V_{max}. \tag{68}$$

Plugging all the pieces together proves the theorem. □

Proof. We first apply Proposition 2 with $M_1 = M^*, M_2 = \widehat{M}, \pi_1 = \pi$, and $\pi_2 = \pi_b$:

$$V^{\pi}_{M^*} - V^{\pi_b}_{M^*} = V^{\pi}_{\widehat{M}} - V^{\pi_b}_{\widehat{M}} + Q^{\pi}_{\widehat{M}}(\pi - \pi_b)\left(d^{\pi_b}_{M^*} - d^{\pi_b}_{\widehat{M}}\right) + \left(Q^{\pi}_{M^*} - Q^{\pi}_{\widehat{M}}\right)(\pi - \pi_b)\, d^{\pi_b}_{M^*} \tag{69}$$

$$V^{\pi}_{M^*}(x) - V^{\pi_b}_{M^*}(x) \leq V^{\pi}_{\widehat{M}}(x) - V^{\pi_b}_{\widehat{M}}(x) - \left\|Q^{\pi}_{\widehat{M}}(\pi - \pi_b)\left(d^{\pi_b}_{M^*} - d^{\pi_b}_{\widehat{M}}\right)(\cdot\,|x)\right\|_1 - \left\|\left(Q^{\pi}_{M^*} - Q^{\pi}_{\widehat{M}}\right)(\pi - \pi_b)\, d^{*}_{\pi_b}(\cdot\,|x)\right\|_1. \tag{70}$$

Then, we apply twice Holder's inequality to the second term 1-norm:

$$\left\|Q^{\pi}_{\widehat{M}}(\pi-\pi_b)\left(d^{\pi_b}_{M^*}-d^{\pi_b}_{\widehat{M}}\right)(\cdot\,|x)\right\|_1$$

$$\leq \left\|Q^{\pi}_{\widehat{M}}\right\|_\infty \left\|(\pi-\pi_b)(\cdot\,|x')\right\|_1 \left\|\left(d^{\pi_b}_{M^*}-d^{\pi_b}_{\widehat{M}}\right)(\cdot\,|x)\right\|_1 \tag{71}$$

$$\leq 2\left\|\left(d^{\pi_b}_{M^*}-d^{\pi_b}_{\widehat{M}}\right)(\cdot\,|x)\right\|_1 . V_{max}. \tag{72}$$

Similarly, we obtain the following with the third term:

$$\left\|\left(Q^{\pi}_{M^*}-Q^{\pi}_{\widehat{M}}\right)(\pi-\pi_b)\, d^*_{\pi_b}(\cdot\,|x)\right\|_1$$

$$\leq \left\|d^*_{\pi_b}(\cdot\,|x)\right\|_1 \max_{x'}\sum_{a'}\left(Q^{\pi}_{M^*}(x',a')-Q^{\pi}_{\widehat{M}}(x',a')\right)(\pi(a'|x')-\pi_b(a'|x')) \tag{73}$$

$$\leq \frac{1}{1-\gamma}\max_{x'}\sum_{a'}\left|Q^{\pi}_{M^*}(x',a')-Q^{\pi}_{\widehat{M}}(x',a')\right||\pi(a'|x')-\pi_b(a'|x')|. \tag{74}$$

We use Lemma 1 to bound the third term in function of $V_{max}, \epsilon, \gamma$, and $e_P(x',a')$:

$$\left\|\left(Q^{\pi}_{M^*}-Q^{\pi}_{\widehat{M}}\right)(\pi-\pi_b)\, d^*_{\pi_b}(\cdot\,|x)\right\|_1$$

$$\leq \frac{1}{1-\gamma}\max_{x'}\sum_{a'}\left(\frac{e_P(x',a')}{1-\kappa\gamma}+\frac{\gamma\epsilon}{(1-\gamma)(1-\kappa\gamma)}\right)V_{max}|\pi(a'|x')-\pi_b(a'|x')| \tag{75}$$

$$\leq \frac{1}{1-\gamma}\max_{x'}\left(\frac{\sum_{a'}e_P(x',a')|\pi(a'|x')-\pi_b(a'|x')|}{1-\kappa\gamma}+\frac{2\gamma\epsilon}{(1-\gamma)(1-\kappa\gamma)}\right)V_{max} \tag{76}$$

$$\leq \frac{1}{1-\gamma}\max_{x'}\frac{1+\gamma}{(1-\gamma)(1-\kappa\gamma)}\ \epsilon V_{max} \tag{77}$$

$$\leq \frac{1+\gamma}{(1-\gamma)^2(1-\kappa\gamma)}\ \epsilon V_{max} \tag{78}$$

where the third line uses the (π_b, e_P, ϵ)-constrained assumption. Plugging all the pieces together proves the theorem. □

Proposition 3. *Let M be an MDP: $\langle \mathcal{X}, \mathcal{A}, P, R, \gamma\rangle$, π_b be the baseline policy on which trajectories $\mathcal{D}$ have been collected, $\widehat{M}$ be the MLE MDP: $\langle \mathcal{X}, \mathcal{A}, P, R, \gamma\rangle$, and $N_{\mathcal{D}}(x)$ be the count of transitions starting from state $x \in \mathcal{X}$ in $\mathcal{D}$. Then, the following concentration bound holds with high probabilities $1-\delta$:*

$$\left\|\left(d^{\pi_b}_{M}-d^{\pi_b}_{\widehat{M}}\right)(\cdot\,|x)\right\|_1 \leq \frac{1}{1-\gamma}\sqrt{\frac{2}{N_{\mathcal{D}}(x)}\log\frac{2^{|\mathcal{X}|}}{\delta}}. \tag{79}$$

Proof. We apply concentration bounds. The proof relies on Theorem 2.1 in Weissman et al. (2003) and is identical to that of Equation 23 borrowed from Petrik et al. (2016), except that $d_M^{\pi_b}$ and $d_{\widehat{M}}^{\pi_b}$ are state distributions and not probability distributions and that their sums are therefore bounded by $\frac{1}{1-\gamma}$. It gives:

$$\left\| \left(d_M^{\pi_b} - d_{\widehat{M}}^{\pi_b} \right) (\cdot|x) \right\|_1 \leq \frac{1}{1-\gamma} \sqrt{\frac{2}{N_{\mathcal{D}}(x)} \log \frac{2^{|\mathcal{X}|}}{\delta}}, \tag{80}$$

which concludes the proof. □

A.5 Proof of Theorem 3

Theorem 3. *The policy improvement step of Approx-Soft-SPIBB generates policies that are guaranteed to be (π_b, e, ϵ)-constrained.*

Proof. The summation of the product of probability mass moved and the error function is below the budget ϵ by design. □

A.6 Proof of Theorem 4

Theorem 4. *Approx-Soft-SPIBB policy improvement has a complexity of $\mathcal{O}(|\mathcal{X}||\mathcal{A}|^2)$.*

Proof. Approx-Soft-SPIBB performs a loop over the state space $\mathcal{X}$ and within that loop a set of operations over the action space. It sorts once the action space in terms of $Q_{\widehat{M}}^{(i)}$ and for each action performs an argmax over $|\mathcal{A}|$. This results in a $|\mathcal{A}|^2$ complexity for each state. □

A.7 Proof of Theorem 5

Theorem 5. *Considering an MDP with exact counts, the model-based policy iteration of (Exact or Approx)-Soft-SPIBB is identical to the model-free policy iteration of (resp. Exact or Approx)-Soft-SPIBB.*

Proof. Let $\pi^{(i+1)}$ be the (exact or approximate) policy improvement, which is assumed to be identical in the model-free and the model-based implementations of Soft-SPIBB. Then, it means that:

$$Q_{M\text{-}Based}^{(i+1)}(x,a) = \widehat{R}(x,a) + \gamma \sum_{x' \in \mathcal{X}} \widehat{P}(x'|x,a) \sum_{a' \in \mathcal{A}} \pi^{(i+1)}(a'|x') Q^{(i)}(x',a') \tag{81}$$

$$= \frac{\sum_{\langle x_j = x, a_j = a, x'_j, r_j \rangle \in \mathcal{D}} r_j}{N_{\mathcal{D}}(x,a)} + \gamma \frac{\sum_{\langle x_j = x, a_j = a, x'_j, r_j \rangle \in \mathcal{D}} \sum_{a' \in \mathcal{A}} \pi^{(i+1)}(a'|x'_j) Q^{(i)}(x'_j, a')}{N_{\mathcal{D}}(x,a)} \tag{82}$$

$$= \frac{y_j^{(i)}}{N_{\mathcal{D}}(x,a)} = Q_{M\text{-}Free}^{(i+1)}(x,a), \tag{83}$$

where Equation 82 is obtained from the MLE MDP $\widehat{M} = \langle \mathcal{X}, \mathcal{A}, \widehat{P}, \widehat{R}, \gamma \rangle$ definition, and Equation 83 is obtained thanks to the target definition from Equation 16. □

A.8 Approximate policy improvement pseudo-code

Pseudo-code 2: Approx-Soft-SPIBB: Policy improvement

Input: Baseline policy π_b, current state x, action set $\mathcal{A}$, errors $e(x,a)$ for each $a \in \mathcal{A}$, precision level ϵ and last iteration value function $Q^{(i)}_{\widehat{M}}$

Initialize $\pi^{(i+1)}(\cdot|x) = \pi_b(\cdot|x)$ and $E = \epsilon$.

Define $\mathcal{A}^-$ as $\mathcal{A}$ sorted in increasing order of $Q^{(i)}_{\widehat{M}}(x,\cdot)$.

for $a^- \in \mathcal{A}^-$ **do**

$$m^- = \min\left(\pi^{(i+1)}(a^-|x), \frac{E}{2e(x,a^-)}\right)$$

$$a^+ = \underset{a \in \mathcal{A}}{\operatorname{argmax}} \frac{Q^{(i)}_{\widehat{M}}(x,a) - Q^{(i)}_{\widehat{M}}(x,a^-)}{e(x,a)}$$

$$m^+ = \min\left(m^-, \frac{E}{2e(x,a^+)}\right)$$

if $m^+ > 0$ **then**

$$\pi^{(i+1)}(a^+|x) = \pi^{(i+1)}(a^+|x) + m^+$$
$$\pi^{(i+1)}(a^-|x) = \pi^{(i+1)}(a^-|x) - m^+$$
$$m^- = m^- - m^+$$
$$E = E - m^+ \left(e(x,a^+) + e(x,a^-)\right)$$

return $\underset{\pi \in \left\{\pi^{(i)}, \pi^{(i+1)}\right\}}{\operatorname{argmax}} \sum_{a \in \mathcal{A}} \pi(a|x) Q^{(i)}_{\widehat{M}}(x,a)$

B Random MDPs benchmark design

B.1 Experiment details

We describe the full experimental process in Pseudo-code 3.

Pseudo-code 3: Random MDPs benchmark

```
Input: List of hyper-parameter values for the baseline
Input: List of dataset size
Input: List of algorithms in the benchmark
Input: List of hyper-parameter values for each algorithm

repeat 10^5 times
    Generate an MDP. (see Section B.1)
    for each hyper parameter value for the baseline do
        Generate a baseline. (see Section B.1)
        for each dataset size do
            Generate a dataset. (see Section B.1)
            for each algorithm do
                for each algorithm hyper-parameter value do
                    Train a policy.
                    Evaluate the policy. (see Section B.1)
                    Record the performance of the trained policy on this run.
```

MDP generation See (Laroche et al., 2019, Appendix B.1.3). Note that in our work, as well as in Laroche et al. (2019), the number of states is set to 50.

Baseline generation See (Laroche et al., 2019, Appendix B.1.4).

Dataset generation The MDP is modified to include another goal: terminal state with a reward of 1 when accessing it. The resulting environment is M^*. We do so to demonstrate the fact that Soft-SPIBB is less conservative than SPIBB, but still safe. The dataset generation depends a single parameter $|\mathcal{D}| \in \{10, 20, 50, 100, 200, 500, 1000, 2000\}$: its size expressed in the number of trajectories. A trajectory generation simply consists in sampling the environment and the baseline policy until reaching the final state. The output is the dataset $\mathcal{D}$.

Policy iteration stopping criterion The policy iteration process stops at iteration i if the Frobenius norm between $Q^{(i)}_{\widehat{M}}$ and $Q^{(i-1)}_{\widehat{M}}$ is lower than 10^{-10}. We did not specify a maximum number of iterations.

Trained policy evaluation See (Laroche et al., 2019, Appendix B.1.6).

Mean and CVaR performance See (Laroche et al., 2019, Appendix B.1.7).

Figures We recall (Laroche et al., 2019, Appendix B.1.8) for clarity of this paper. We present three types of figure in the paper (main document and appendix).

Performance vs. dataset size: These figures (for instance Figure 2(a)) show the (mean and/or CVaR) performance of the algorithms as a function of the dataset size.

Hyper-parameter search heatmaps: These figures (for instance Figure 6(a)) show the (mean or CVaR) normalized performance of the algorithms as a function of both the dataset size and the hyper-parameter value of the evaluated algorithm. The normalized performance is computed with Equation 17 and represented with colour. Red means that the performance is worse than that of the baseline, yellow means that it is equal and green means that it improves the baseline.

Random MDPs heatmaps: These figures (for instance Figure 6(e)) are very similar to the other heatmaps except that the normalized performance is shown as a function of both the dataset size and the hyper-parameter η used for the baseline generation (instead of the hyper-parameter of the evaluated algorithm).

B.2 Other benchmark algorithms: competitors

Since the *baseline* meaning is overridden in this paper, we refer to the non-Soft-SPIBB benchmark algorithms with the term *competitors*.

Basic RL, HCPI, Robust MDP, Reward-adjusted MDP See (Laroche et al., 2019, Appendix B.2.1 to B.2.4).

SPIBB algorithms The theory developed in Laroche et al. (2019) first computes from the dataset $\mathcal{D}$, the bootstrapped set $\mathfrak{B}$ of the state-action pairs that were not sampled enough to allow a good estimate of the model. The construction of $\mathfrak{B}$ depends on a hyper-parameter $N_\wedge$ which defines the minimal number of samples for a state-action pair to be considered well-known. Then, based on $\mathfrak{B}$, two algorithms are considered:

- The safety-proven Π_b-SPIBB copies the baseline policy for state-action pairs in $\mathfrak{B}$ and greedily optimizes in the complementary state-action pairs set.
- The empirically-improved $\Pi_{\leq b}$-SPIBB copies the baseline policy for state-action pairs in $\mathfrak{B}$ and greedily optimizes in the complementary state-action pairs set.

Table 1 illustrates the difference between Π_b-SPIBB and $\Pi_{\leq b}$-SPIBB in the policy improvement step of the policy iteration process. It shows how the baseline probability mass is locally redistributed among the different actions for the two policy-based SPIBB algorithms. We observe that for Π_b-SPIBB, the bootstrapped state-action pairs probabilities remain untouched whatever their Q-value estimates are. On the contrary,

$\Pi_{\leq b}$-SPIBB removes all mass from the bootstrapped state-action pairs that are performing worse than the current Q-value estimates.

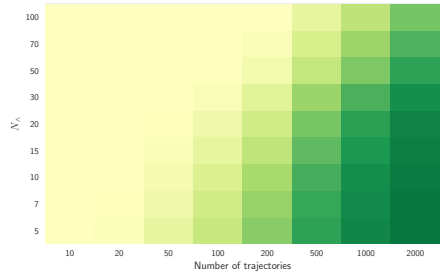
Figure 6 show the result of the hyper-parameter search. The mean performance improves as the safety decreases. We choose $N_\wedge = 10$ in our experiments because it seems to offer the best compromise and Soft-SPIBB turns out the improve at the same time SPIBB safety and mean performance under this setting.

B.3 More Random MDPs experiment results

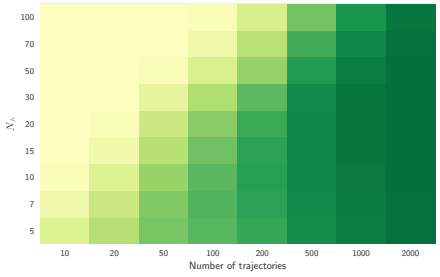
We provide next more raw results of our experiments in an arbitrary (not random) order: see Figures 7, 8, 9, 10 and 11.

Table 1. Policy improvement step at iteration (i) for the two policy-based SPIBB algorithms.

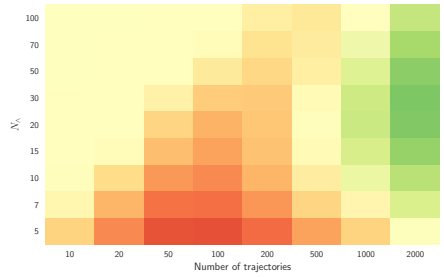
Q-value estimate	Baseline policy	Boostrapping	Π_b-SPIBB	$\Pi_{\leq b}$-SPIBB
$Q^{(i)}_{\widehat{M}}(x, a_1) = 1$	$\pi_b(a_1\|x) = 0.1$	$(x, a_1) \in \mathfrak{B}$	$\pi^{(i+1)}(a_1\|x) = 0.1$	$\pi^{(i+1)}(a_1\|x) = 0$
$Q^{(i)}_{\widehat{M}}(x, a_2) = 2$	$\pi_b(a_2\|x) = 0.4$	$(x, a_2) \notin \mathfrak{B}$	$\pi^{(i+1)}(a_2\|x) = 0$	$\pi^{(i+1)}(a_2\|x) = 0$
$Q^{(i)}_{\widehat{M}}(x, a_3) = 3$	$\pi_b(a_3\|x) = 0.3$	$(x, a_3) \notin \mathfrak{B}$	$\pi^{(i+1)}(a_3\|x) = 0.7$	$\pi^{(i+1)}(a_3\|x) = 0.8$
$Q^{(i)}_{\widehat{M}}(x, a_4) = 4$	$\pi_b(a_4\|x) = 0.2$	$(x, a_4) \in \mathfrak{B}$	$\pi^{(i+1)}(a_4\|x) = 0.2$	$\pi^{(i+1)}(a_4\|x) = 0.2$



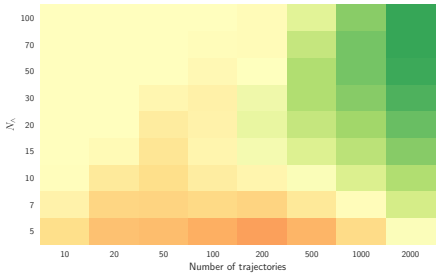
(a) Mean Π_b-SPIBB with $\eta = 0.9$



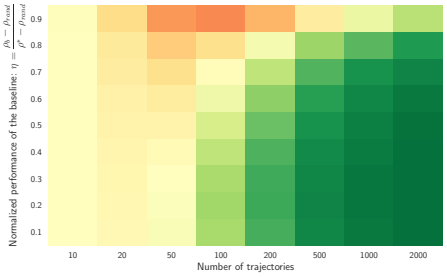
(b) Mean $\Pi_{\leq b}$-SPIBB with $\eta = 0.9$



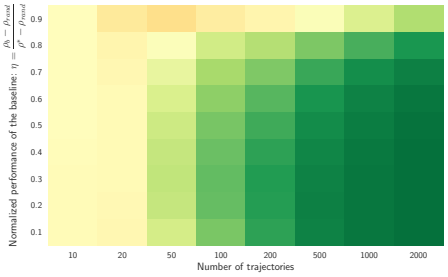
(c) 1%-CVaR Π_b-SPIBB with $\eta = 0.9$



(d) 1%-CVaR $\Pi_{\leq b}$-SPIBB with $\eta = 0.9$

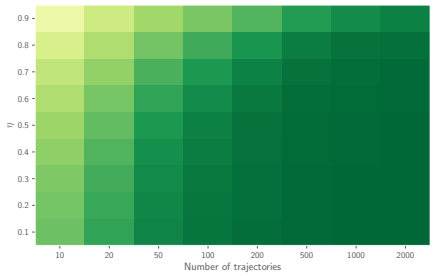


(e) 1%-CVaR Π_b-SPIBB ($N_\wedge = 10$)

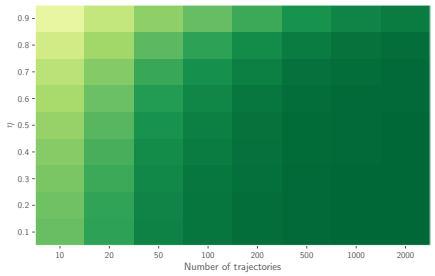


(f) 1%-CVaR $\Pi_{\leq b}$-SPIBB ($N_\wedge = 10$)

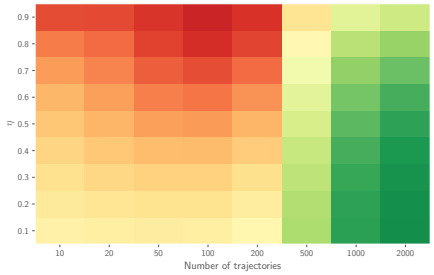
Fig. 6. Random MDPs (no additional goal): SPIBB hyper-parameter search results: (a-d) Mean and 1%-CVaR performance heatmaps as a function of $N_\wedge$ (e-f) 1%-CVaR performance heatmaps as a function of η with the best hyper-parameter ($N_\wedge = 10$).



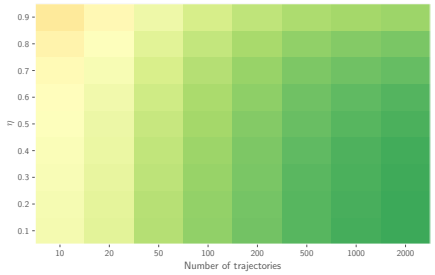
(a) Mean: Basic RL.



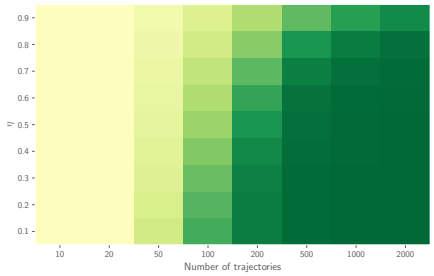
(b) Mean: RaMDP.



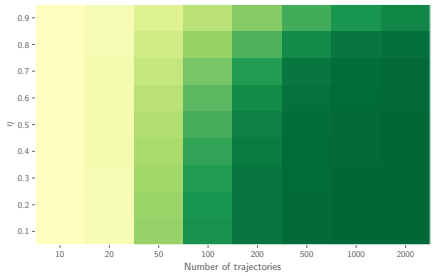
(c) Mean: Robust MDP.



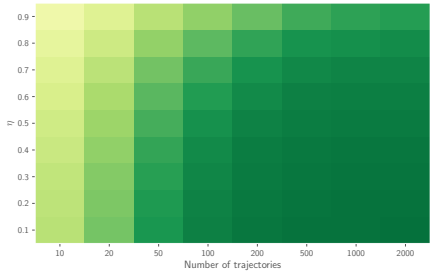
(d) Mean: HCPI doubly robust.



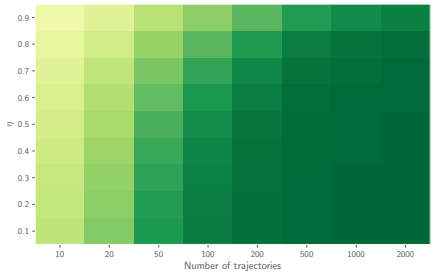
(e) Mean: Π_b-SPIBB.



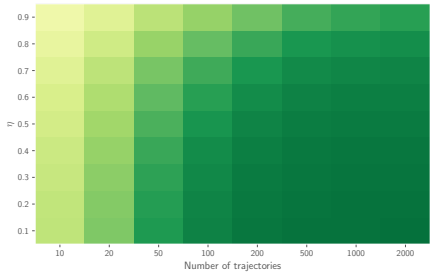
(f) Mean: $\Pi_{\leq b}$-SPIBB.



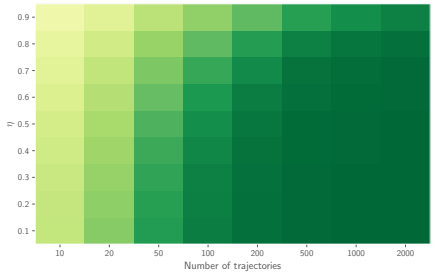
(g) Mean: Exact-Soft-SPIBB 1-step ($\epsilon = 2$).



(h) Mean: Exact-Soft-SPIBB ($\epsilon = 2$).

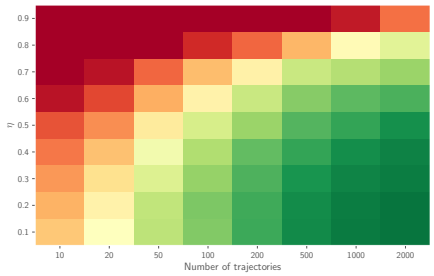


(i) Mean: Approx-Soft-SPIBB 1-step ($\epsilon = 2$).

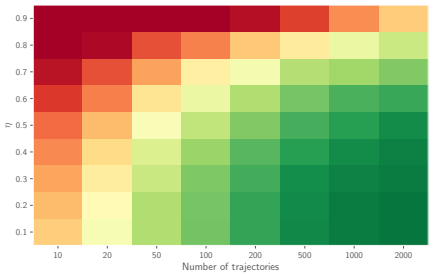


(j) Mean: Approx-Soft-SPIBB ($\epsilon = 2$).

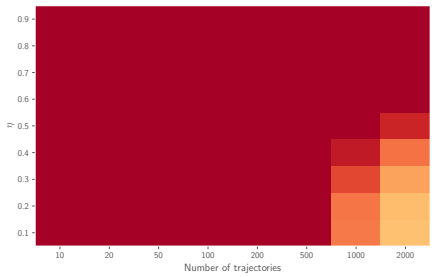
Fig. 7. Random MDPs: mean performance heatmaps.



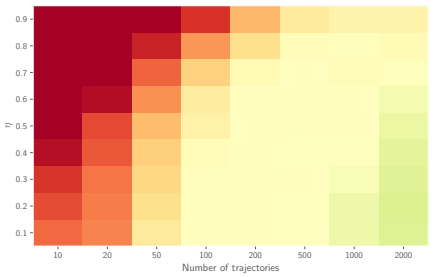
(a) 0.1%-CVaR: Basic RL.



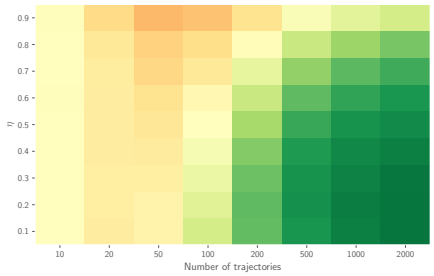
(b) 0.1%-CVaR: RaMDP.



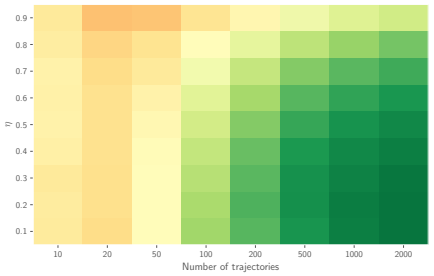
(c) 0.1%-CVaR: Robust MDP.



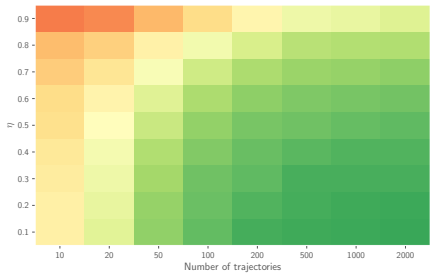
(d) 0.1%-CVaR: HCPI doubly robust.



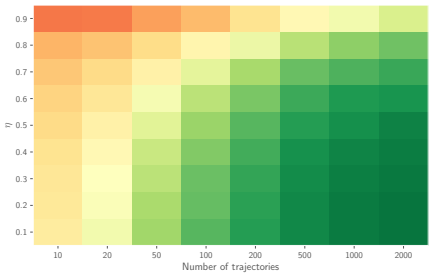
(e) 0.1%-CVaR: Π_b-SPIBB.



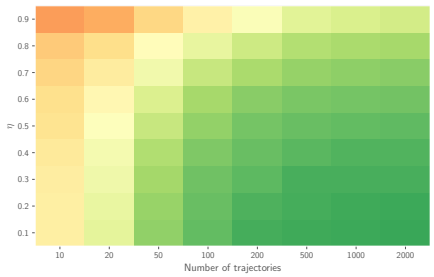
(f) 0.1%-CVaR: $\Pi_{\leq b}$-SPIBB.



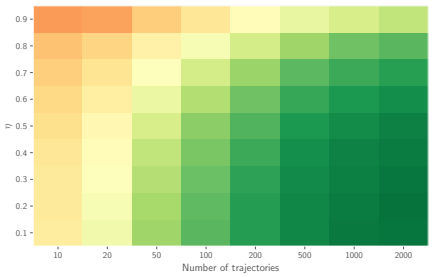
(g) 0.1%-CVaR: Exact-Soft-SPIBB 1-s ($\epsilon = 2$).



(h) 0.1%-CVaR: Exact-Soft-SPIBB ($\epsilon = 2$).

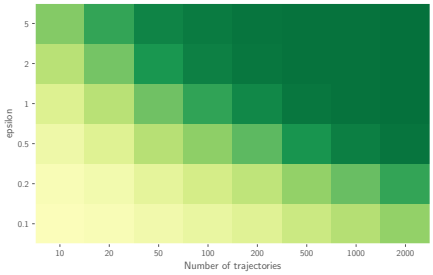


(i) 0.1%-CVaR: Approx-Soft-SPIBB 1-s ($\epsilon = 2$).

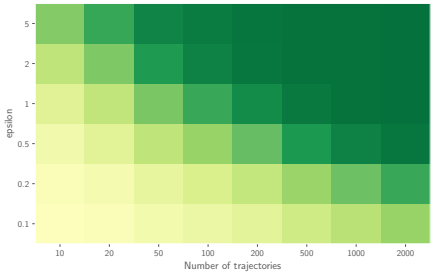


(j) 0.1%-CVaR: Approx-Soft-SPIBB ($\epsilon = 2$).

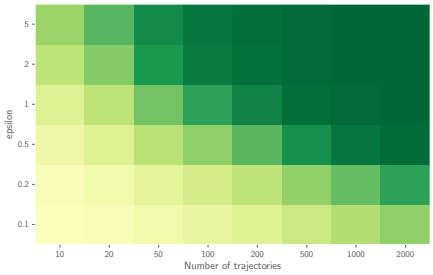
Fig. 8. Random MDPs: 0.1%-CVaR performance heatmaps.



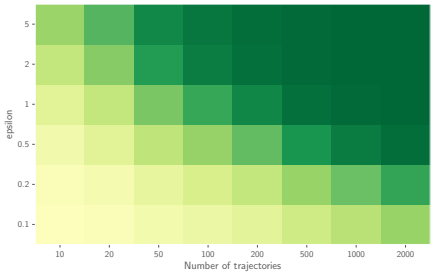
(a) Mean: Exact-Soft-SPIBB 1-step ($\eta = 0.1$).



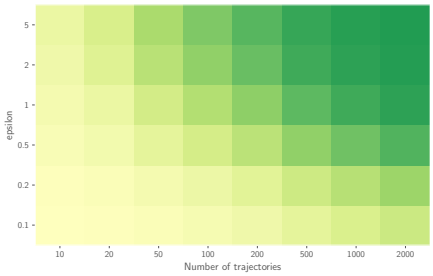
(b) Mean: Approx-Soft-SPIBB 1-step ($\eta = 0.1$).



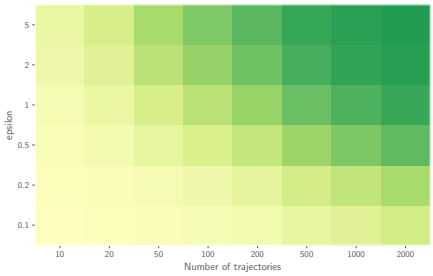
(c) Mean: Exact-Soft-SPIBB ($\eta = 0.1$).



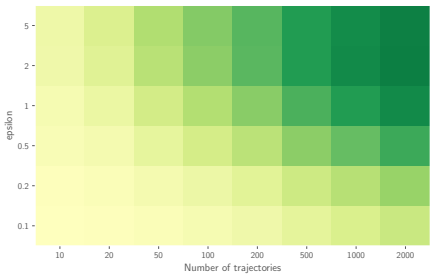
(d) Mean: Approx-Soft-SPIBB ($\eta = 0.1$).



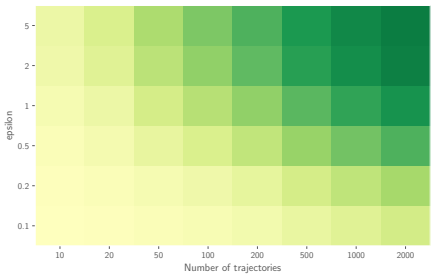
(e) Mean: Exact-Soft-SPIBB 1-step ($\eta = 0.9$).



(f) Mean: Approx-Soft-SPIBB 1-step ($\eta = 0.9$).

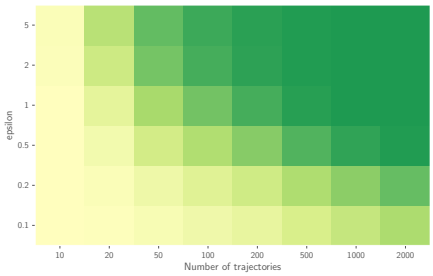


(g) Mean: Exact-Soft-SPIBB ($\eta = 0.9$).

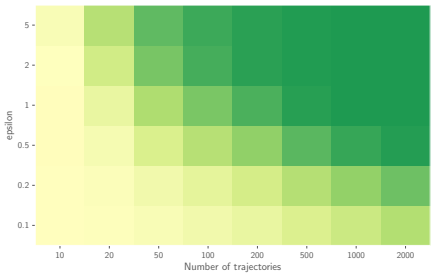


(h) Mean: Approx-Soft-SPIBB ($\eta = 0.9$).

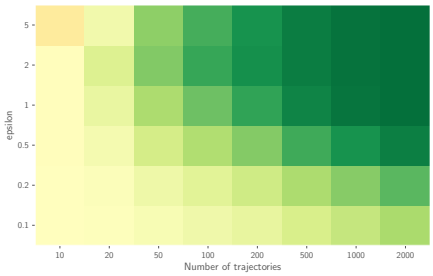
Fig. 9. Random MDPs: hyper-parameter mean performance heatmaps for Soft-SPIBB methods under a weak ($\eta = 0.1$) and a strong ($\eta = 0.9$) baseline.



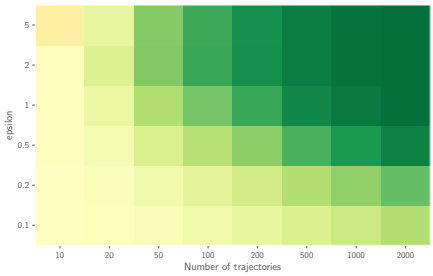
(a) 1%-CVaR: Exact-Soft-SPIBB 1-s ($\eta = 0.1$).



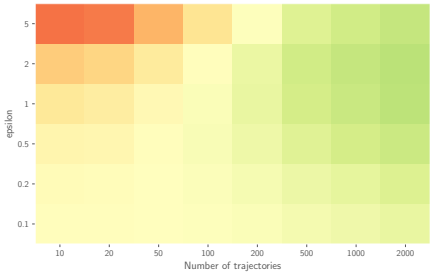
(b) 1%-CVaR: Approx-Soft-SPIBB 1-s ($\eta = 0.1$).



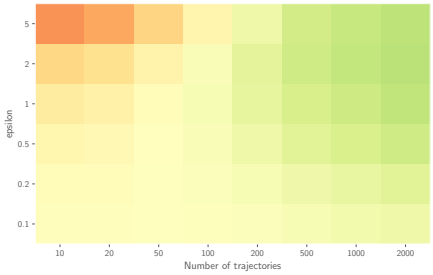
(c) 1%-CVaR: Exact-Soft-SPIBB ($\eta = 0.1$).



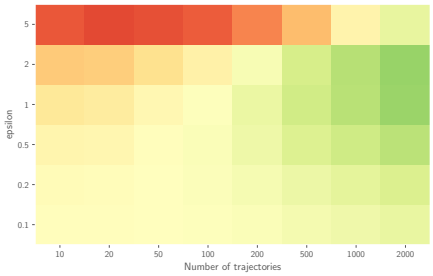
(d) 1%-CVaR: Approx-Soft-SPIBB ($\eta = 0.1$).



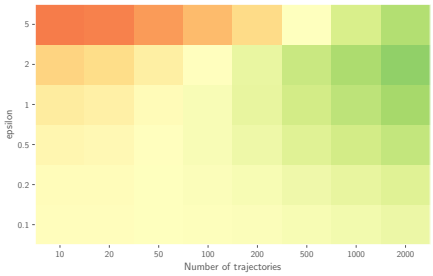
(e) 1%-CVaR: Exact-Soft-SPIBB 1-s ($\eta = 0.9$).



(f) 1%-CVaR: Approx-Soft-SPIBB 1-s ($\eta = 0.9$).

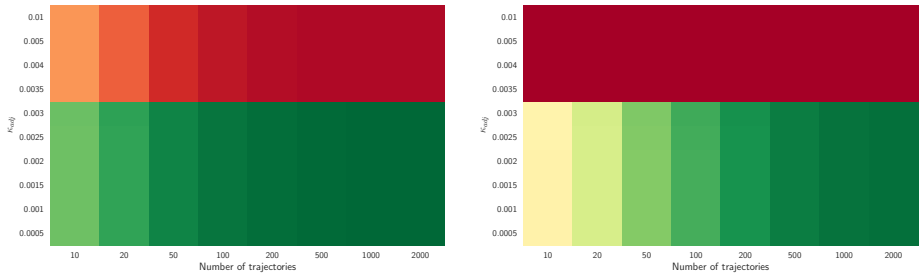


(g) 1%-CVaR: Exact-Soft-SPIBB ($\eta = 0.9$).



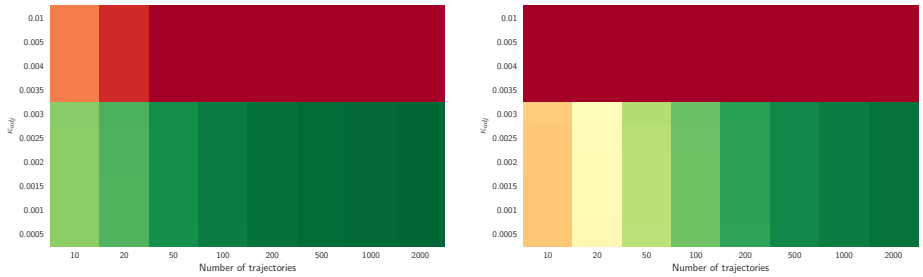
(h) 1%-CVaR: Approx-Soft-SPIBB ($\eta = 0.9$).

Fig. 10. Random MDPs: hyper-parameter 1%-CVaR performance heatmaps for Soft-SPIBB methods under a weak ($\eta = 0.1$) and a strong ($\eta = 0.9$) baseline.



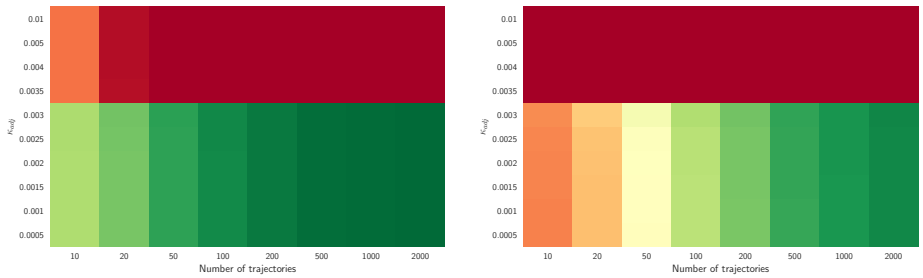
(a) Mean: RaMDP ($\eta = 0.1$).

(b) 1%-CVaR: RaMDP ($\eta = 0.1$).



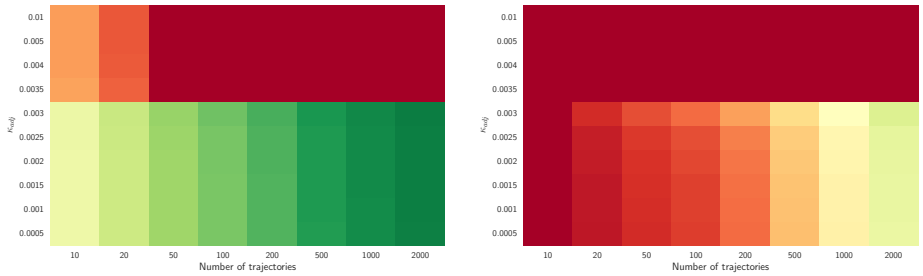
(c) Mean: RaMDP ($\eta = 0.4$).

(d) 1%-CVaR: RaMDP ($\eta = 0.4$).



(e) Mean: RaMDP ($\eta = 0.6$).

(f) 1%-CVaR: RaMDP ($\eta = 0.6$).



(g) Mean: RaMDP ($\eta = 0.9$).

(h) 1%-CVaR: RaMDP ($\eta = 0.9$).

Fig. 11. Random MDPs: hyper-parameter mean and 1%-CVaR performance heatmaps for RaMDP methods under a weak ($\eta = 0.1$), a medium weak ($\eta = 0.4$), a medium strong ($\eta = 0.6$), and a strong ($\eta = 0.9$) baseline.

C Helicopter experiment details

C.1 Details about the helicopter environment

See (Laroche et al., 2019, Appendix D.1).

C.2 Helicopter experimental process

See Pseudo-code 4.

Pseudo-code 4: Helicopter experimental process

```
Input: List of algorithms
Input: List of hyper-parameter values
Input: List of dataset sizes

repeat 20 times
    for each dataset size do
        Generate a dataset.
        Compute the pseudo-counts.
        repeat 15 times
            for each algorithm do
                for each hyper-parameter value do
                    Train a policy.
                    Evaluate the trained policy.
                    Record the performance of the trained policy.
```

C.3 Details about the DQN implementations

The batch version of DQN simply consists in replacing the experience replay buffer by the dataset we are training on. Effectively, we are not sampling from the environment anymore but from the transitions collected a priori following the baseline. RaMDP follows the same principle with a target modified according to the reward hacking:

$$y_j^{(i)} = r_j - \frac{\kappa}{\sqrt{\tilde{N}_{\mathcal{D}}(x_j, a_j)}} + \gamma \max_{a' \in \mathcal{A}} Q^{(i)}(x'_j, a'), \tag{84}$$

where $\tilde{N}_{\mathcal{D}}(x_j, a_j)$ is the pseudo-count. For the sake of simplicity and to be able to repeat several runs of each experiment efficiently, instead of applying pseudo-count methods from the literature (Bellemare et al., 2016; Fox et al., 2018; Burda et al., 2019), we consider here a pseudo-count heuristic based on the Euclidean state-distance, and a task where it makes sense to do so. The pseudo-count of a state-action (x, a) is defined as the sum of its similarity with the state-action pairs (x_i, a_i) found in the dataset.

The similarity between (x, a) and (x_i, a_i) is equal to 0 if $a_i \neq a$, and to $\max(0, 1 - 5d(x, x_i))$ otherwise, where $d(\cdot, \cdot)$ is the Euclidean distance between two states:

$$\tilde{N}_{\mathcal{D}}(x, a) = \sum_{\langle x_j, a_j = a, r_j, x'_j \rangle \in \mathcal{D}} \max(0, 1 - 5d(x, x_j)), \tag{85}$$

The same pseudo-counts were used for all algorithms using it. In particular, they allowed to compute the error function at each state-action pair as (the cardinal of continuous state spaces cannot be used as in the tabular case):

$$e_P(x, a) = \frac{1}{\sqrt{\tilde{N}_{\mathcal{D}}(x, a)}}. \tag{86}$$

The same general methodology applies for Soft-SPIBB, according to the definitions from Section 3.2. Although, SPIBB is a special case of Soft-SPIBB, its implementation has been performed independently in order to be more computationally efficient. For SPIBB, the targets we are using for our Q-values updates verify the following modified Bellman equation:

$$\begin{aligned} y_j^{(i)} &= r_j + \gamma \max_{\pi \in \Pi_b} \sum_{a' \in \mathcal{A}} \pi(a'|x'_j) Q^{(i)}(x'_j, a') && (87) \\ &= r_j + \gamma \sum_{(x'_j, a') \in \mathfrak{B}} \pi_b(a'|x'_j) Q^{(i)}(x'_j, a') \\ &\quad + \gamma \left(\sum_{(x'_j, a') \notin \mathfrak{B}} \pi_b(a'|x'_j) \right) \max_{(x'_j, a') \notin \mathfrak{B}} Q^{(i)}(x'_j, a') && (88) \end{aligned}$$

We notice in particular that when $\mathfrak{B} = \emptyset$ the targets fall back to the traditional Bellman ones. We used the now classic target network trick (Mnih et al., 2015), combined with Double-DQN (van Hasselt et al., 2015).

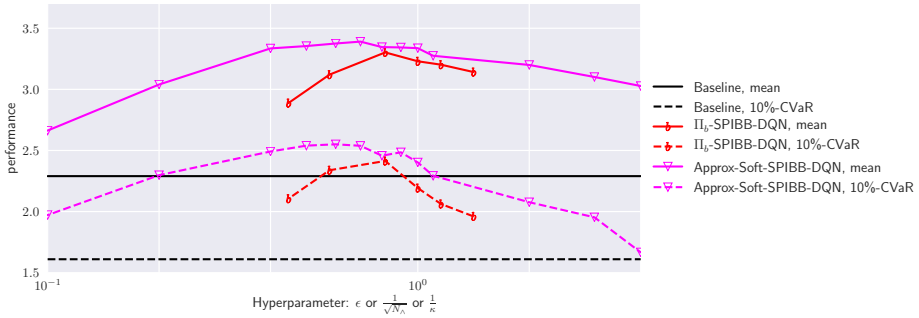
The network used for the baseline and for the algorithms in the benchmark is a fully connected network with 3 hidden layers of 32, 128 and 32 neurons, initialized using he_uniform (He et al., 2015). The network has 9 outputs corresponding to the Q-values of the 9 actions in the game. We train the Q-networks with RMSProp (Tieleman and Hinton, 2012) with a momentum of 0.95 and $\epsilon = 10^{-7}$ on mini-batches of size 32. The learning rate is initialized at 0.01 and is annealed every 20k transitions or every pass on the dataset, whichever is larger. The networks are trained for 2k passes on the dataset, and are fully converged by that time. The models are trained with Pytorch (Paszke et al., 2017). The policy is tested for 1k steps at the end of training, with the initial states of each trajectory sampled as described in Section C.1.

C.4 Additional experimental results

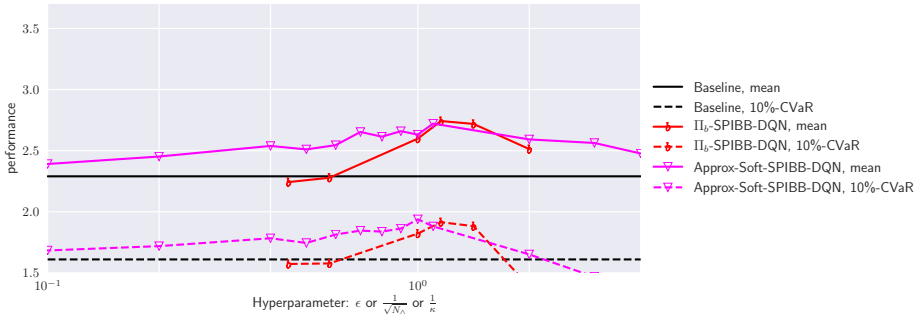
See Table 2 and Figure 12.

Table 2. Helicopter: numerical (mean and 10%-CVaR) results with optimal hyper-parameters

$\lvert\mathcal{D}\rvert$	Baseline		DQN		RaMDP-DQN		Π_b-SPIBB		Soft-SPIBB	
	Mean	CVaR	Mean	CVaR	Mean	CVaR	Mean	CVaR	Mean	CVaR
1,000	2.30	1.59	-0.91	-1.00	N/A	N/A	**2.74**	**1.91**	**2.72**	**1.88**
3,000	2.30	1.59	-0.72	-1.00	0.85	-0.70	**3.19**	**2.37**	**3.23**	**2.38**
6,000	2.30	1.59	0.07	-1.00	N/A	N/A	3.30	2.41	**3.39**	**2.54**
10,000	2.30	1.59	0.22	-1.00	3.21	1.65	3.39	2.46	**3.61**	**2.69**



(a) Helicopter benchmark with $\lvert\mathcal{D}\rvert = 6,000$



(b) Helicopter benchmark with $\lvert\mathcal{D}\rvert = 1,000$

Fig. 12. Helicopter: mean and 10%-CVaR as a function of the hyper-parameter

D Reproducible, reusable, and robust Reinforcement Learning

This paper's objective is to improve the robustness and the reliability of Reinforcement Learning algorithms. Inspired from Joelle Pineau's talk at NeurIPS 2018 about reproducible, reusable, and robust Reinforcement Learning[1], we intend to also make our work reusable and reproducible.

D.1 Pineau's checklist (slide 33)

For all algorithms presented, check if you include:

- A clear description of the algorithm.
  ⇒ See Section 3.
- An analysis of the complexity (time, space, sample size) of the algorithm.
  ⇒ We provide formal analysis for the complexity in Section 3.2, and a rapid empirical assessment in Figure 1 for the finite MDP analysis. For Soft-SPIBB-DQN, we empirically observed a 50% increase in runtime compared to SPIBB-DQN.
- A link to downloadable source code, including all dependencies.
  ⇒ The code is being made available. The code will be made public on acceptation. See Section D.2.

For any theoretical claim, check if you include:

- A statement of the result.
  ⇒ See Theorems 1, 2, and 4.
- A clear explanation of any assumptions.
  ⇒ The main assumption for Theorem 2 is formalized as Assumption 1. See Section 3 for discussion.
- A complete proof of the claim.
  ⇒ See Section A.

For all figures and tables that present empirical results, check if you include:

- A complete description of the data collection process, including sample size.
  ⇒ See Sections 4, B.1, and C.1.
- A link to downloadable version of the dataset or simulation environment.
  ⇒ See Section D.2.
- An explanation of how sample were allocated for training / validation / testing.
  ⇒ The complete dataset is used for training. There is no need for validation set. Testing is performed in the true environment.
- An explanation of any data that was excluded.
  ⇒ Does not apply to our simulated environments.
- The range of hyper-parameters considered, method to select the best hyper-parameter configuration, and specification of all hyper-parameters used to generate results.
  ⇒ See Sections B.2 and C.3.

[1] https://nips.cc/Conferences/2018/Schedule?showEvent=12486

- The exact number of evaluation runs.
  ⇒ 100,000+ for finite MDPs experiments and 300 for SPIBB-DQN experiments.
- A description of how experiments were run.
  ⇒ See Sections 4, B, and C.
- A clear definition of the specific measure or statistics used to report results.
  ⇒ Mean and X% conditional value at risk (CVaR), described in Sections 4 and B.1.
- Clearly defined error bars.
  ⇒ Given the high number of runs we considered, the error bar are too thin to be displayed. Any difference visible with the naked eye is significant. We use CVaR everywhere instead to account for the uncertainty.
- A description of results including central tendency (e.g. mean) and variation (e.g. stddev).
  ⇒ All our work is motivated and analyzed with respect to this matter.
- The computing infrastructure used.
  ⇒ For the finite-MDPs experiment, we used clusters of CPUs. The full results were obtained by running the benchmarks with 100 CPUs running independently in parallel during 24h. For the helicopter experiment, we used a GPU cluster. However, only one GPU is necessary for a single run. Using a cluster allowed to launch several runs in parallel and considerably sped up the experiment. On a single GPU (a GTX 1080 Ti), a dataset of $|\mathcal{D}| = 10$k transitions is generated in 5 seconds. The dataset generation scales linearly in $|\mathcal{D}|$. Computing the counts for that dataset takes approximately 20 minutes, it scales quadratically with the size of the dataset. As far as training is concerned, 2000 passes on a dataset of 10k transitions takes around 25 minutes, it scales linearly in N. Finally, evaluation of the trained policy on 10k trajectories takes 15 minutes. It scales linearly in $|\mathcal{D}|$ as it requires the computation of the pseudo-count for each state encountered during the evaluation and this pseudo-count computation is linear in $|\mathcal{D}|$. Overall, a single run for a dataset of 10k transitions takes around one hour.

D.2 Code attached to the submission

The attached code can be used to reproduce the experiments presented in the submitted paper. It is split into two projects: one for finite MDPs, and one for Soft-SPIBB-DQN.

- https://github.com/RomainLaroche/SPIBB
- https://github.com/rems75/SPIBB-DQN

Finite MDPs

Prerequisites The finite MDP project is implemented in Python 3.5 and only requires *numpy* and *scipy*.

Content We include the following:

- Libraries of the following algorithms:
  - Basic RL,
  - Soft-SPIBB:
    - Exact-Soft-SPIBB,
    - Approx-Soft-SPIBB,
  - SPIBB:
    - Π_b-SPIBB,
    - $\Pi_{\leq b}$-SPIBB,
  - HCPI:
    - doubly-robust,
    - importance sampling,
    - weighted importance sampling,
    - weighted per decision IS,
    - per decision IS,
  - Robust MDP,
  - and Reward-adjusted MDP.
- Environments:
  - Random MDPs environment.
- Random MDPs experiment of Section 4.1. Run:
  `python main.py #name_of_experiment# #random_seed#`

Not included We DO NOT include the following:

- The hyper-parameter search (appendix, Section B.2): it should be easy to re-implement.
- The figure generator: it has too many specificities to be made understandable for a user at the moment. Also, it is not hard to re-implement with one's own visualization tools.

License This project is BSD-licensed.

Soft-SPIBB-DQN

Prerequisites Soft-SPIBB-DQN is implemented in Python 3 and requires the following libraries: PyTorch, pickle, glob, yaml, argparse, numpy, yaml, pathlib, csv, scipy and click.

Content The SoftSPIBB-DQN project contains the helicopter environment, the baseline used for our experiments and the code required to generate datasets and train vanilla DQN, RaMDP-DQN, SPIBB-DQN and Soft-SPIBB-DQN.

Not included We DO NOT include the following:

- The multi-CPU/multi-GPU implementation: its structure is too much dependent on the cluster tools. It would be useless for somebody from another lab and might divulge author affiliations.

License This project is BSD-licensed.