On the Performance Bounds of some Policy Search Dynamic Programming Algorithms

Bruno Scherrer,
LORIA – MAIA project-team,
Nancy, France,
bruno.scherrer@inria.fr

June 3, 2013

Abstract

We consider the infinite-horizon discounted optimal control problem formalized by Markov Decision Processes. We focus on Policy Search algorithms, that compute an approximately optimal policy by following the standard Policy Iteration (PI) scheme via an ϵ-approximate greedy operator (Kakade and Langford, 2002; Lazaric *et al.*, 2010). We describe existing and a few new performance bounds for Direct Policy Iteration (DPI) (Lagoudakis and Parr, 2003; Fern *et al.*, 2006; Lazaric *et al.*, 2010) and Conservative Policy Iteration (CPI) (Kakade and Langford, 2002). By paying a particular attention to the concentrability constants involved in such guarantees, we notably argue that the guarantee of CPI is much better than that of DPI, but this comes at the cost of a relative—exponential in $\frac{1}{\epsilon}$—increase of time complexity. We then describe an algorithm, Non-Stationary Direct Policy Iteration (NSDPI), that can either be seen as 1) a variation of Policy Search by Dynamic Programming by Bagnell *et al.* (2003) to the infinite horizon situation or 2) a simplified version of the Non-Stationary PI with growing period of Scherrer and Lesner (2012). We provide an analysis of this algorithm, that shows in particular that it enjoys the best of both worlds: its performance guarantee is similar to that of CPI, but within a time complexity similar to that of DPI.

1 Introduction

The study of approximation in Dynamic Programming algorithms for infinite-horizon discounted Markov Decision Processes (MDP) has a rich history (Bertsekas and Tsitsiklis, 1996; Szepesvári, 2010). Some of the first important results, gathered by Bertsekas and Tsitsiklis (1996), provide bounds on the closeness to optimality of the computed policy as a function of the *max-norm* errors during iterations. If Value or Policy Iteration are run with some error ϵ_k, it is known that the value v_{π_k} of the policies π_k generated by the algorithm can get close to the optimal policy π_* if the errors are small enough since we have

$$\limsup_{k\to\infty} \|v_{\pi_*} - v_{\pi_k}\|_\infty \leq \frac{2\gamma}{(1-\gamma)^2} \sup_k \|\epsilon_k\|_\infty.$$

Unfortunately, such results have a limited range since in practice, most implementations of Dynamic Programming algorithms involve function approximation (like classification or regression) that controls some ν-weighted L_p norm $\|\cdot\|_{\nu,p}$ instead of the max-norm $\|\cdot\|_\infty$. Starting with the works of Munos (2003, 2007), this motivated a recent trend of research (Antos *et al.*, 2008; Munos and Szepesvári, 2008; Farahmand *et al.*, 2009, 2010; Lazaric *et al.*, 2010, 2011; Scherrer *et al.*, 2012) that provide generalizations of the above result of the form

$$\limsup_{k\to\infty} \|v_{\pi_*} - v_{\pi_k}\|_{\mu,p} \leq \frac{2\gamma C^{1/p}}{(1-\gamma)^2} \sup_k \|\epsilon_k\|_{\nu,p}, \tag{1}$$

where μ and ν are some distributions. The possibility to express the right hand side with respect to some ν-weighted L_p norm comes at the price of a constant C, called concentrability coefficient, that measures the stochastic smoothness of the MDP: the more the MDP dynamics may concentrate in some parts of

the state space, the bigger C (see Munos and Szepesvári (2008) for a detailed discussion). Though efforts have been employed to improve these constants (Farahmand *et al.*, 2010; Scherrer *et al.*, 2012), the fact that it may be infinite (for instance when the MDP is deterministic) constitutes a severe limitation.

Interestingly, one work (Kakade and Langford, 2002; Kakade, 2003)—anterior to those of Munos (2003, 2007)—proposed an approximate Dynamic Programming algorithm, Conservative Policy Iteration (CPI), with a performance bounds similar to Equation (1) but with a constant that is—as we will argue precisely in this paper (Remarks 1, 2 and 3)—better than all the others: it only involves the mismatch between some input measure of the algorithm and a baseline distribution corresponding roughly to the frequency visitation of the optimal policy, a natural piece of information that an expert of the domain may provide. The main motivation of this paper is to emphasize the importance of these concentrability constants regarding the significance of the performance bounds.

In Section 3, we will describe Direct Policy Iteration (DPI), a very simple Dynamic Programming algorithm proposed by Lagoudakis and Parr (2003); Fern *et al.* (2006); Lazaric *et al.* (2010) that is similar to CPI; for this algorithm, we will provide and extend the analysis developed by Lazaric *et al.* (2010). We will then consider CPI in Section 4, describe the theoretical properties originally given by Kakade and Langford (2002), as well as some new bounds that will ease the comparison with DPI. In particular, as a corollary of our analysis, we will obtain an original bound for CPI(α), a practical variation of CPI that uses a fixed stepsize. We will argue that the concentrability constant involved in the analysis of CPI is better than those of DPI. This improvement of quality unfortunately comes at some price: the number of iterations required by CPI can be exponentially bigger than that of DPI. This will motivate the introduction of another algorithm: we describe in Section 5 Non-Stationary Direct Policy Iteration (NSDPI), that can either be seen as 1) a variation of *Policy Search by Dynamic Programming* by Bagnell *et al.* (2003) to the infinite horizon situation or 2) a simplification of the *Non-Stationary PI with growing period* algorithm of Scherrer and Lesner (2012). We will analyze this algorithm and prove in particular that it enjoys the best of both worlds: a guarantee similar to that of CPI *and* a fast rate like that of DPI. The next section begins by providing the background and the precise setting considered.

2 Background

We consider an infinite-horizon discounted Markov Decision Process Puterman (1994); Bertsekas and Tsitsiklis (1996) $(\mathcal{S}, \mathcal{A}, P, r, \gamma)$, where $\mathcal{S}$ is a possibly infinite state space, $\mathcal{A}$ is a finite action space, $P(ds'|s,a)$, for all (s,a), is a probability kernel on $\mathcal{S}$, $r : \mathcal{S} \to [-R_{\max}, R_{\max}]$ is a reward function bounded by $R_{\max}$, and $\gamma \in (0,1)$ is a discount factor. A stationary deterministic policy $\pi : \mathcal{S} \to \mathcal{A}$ maps states to actions. We write $P_\pi(ds'|s) = P(ds'|s, \pi(s))$ for the stochastic kernel associated to policy π. The value v_π of a policy π is a function mapping states to the expected discounted sum of rewards received when following π from any state: for all $s \in \mathcal{S}$,

$$v_\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t) \middle| s_0 = s,\ s_{t+1} \sim P_\pi(\cdot|s_t)\right].$$

The value v_π is clearly bounded by $V_{\max} = R_{\max}/(1-\gamma)$. It is well-known that v_π can be characterized as the unique fixed point of the linear Bellman operator associated to a policy π: $T_\pi : v \mapsto r + \gamma P_\pi v$. Similarly, the Bellman optimality operator $T : v \mapsto \max_\pi T_\pi v$ has as unique fixed point the optimal value $v_* = \max_\pi v_\pi$. A policy π is greedy w.r.t. a value function v if $T_\pi v = Tv$, the set of such greedy policies is written $\mathcal{G}v$. Finally, a policy π_* is optimal, with value $v_{\pi_*} = v_*$, iff $\pi_* \in \mathcal{G}v_*$, or equivalently $T_{\pi_*} v_* = v_*$.

In this paper, we focus on algorithms that use an approximate greedy operator, $\mathcal{G}_\epsilon$, that takes as input a distribution ν and a function $v : S \to \mathbb{R}$ and returns a policy π that is (ϵ, ν)-approximately greedy with respect to v in the sense that:

$$\nu(Tv - T_\pi v) = \nu(\max_{\pi'} T_{\pi'} v - T_\pi v) \leq \epsilon.$$

In practice, this can be achieved through a L_1-regression of the so-called *advantage function* Kakade and Langford (2002); Kakade (2003) or through a sensitive classification problem Lazaric *et al.* (2010), in both case generating the learning problems through rollout trajectories induced by policy π. For all

considered algorithms, we will provide bounds on the expected loss $E_{s\sim\mu}[v_{\pi_*}(s) - v_\pi(s)] = \mu(v_{\pi_*} - v_\pi)$ of using some generated policy π instead of the optimal policy π_* for some distribution μ of interest as a function of the errors ϵ_k at each iteration.

3 Direct Policy Iteration (DPI)

We begin by describing Direct Policy Iteration (DPI) introduced by Lagoudakis and Parr (2003); Fern *et al.* (2006) and analyzed by Lazaric *et al.* (2010). The analysis of this algorithm relies on the following

Algorithm 1 DPI

input: an initial policy π_0, a distribution ν
for $k = 0, 1, 2, \dots$ **do**
  $\pi_{k+1} \leftarrow \mathcal{G}_{\epsilon_{k+1}}(\nu, v_{\pi_k})$
end for
return: v_{π_k}

coefficients relating μ (the distribution of interest for measuring the loss) and ν (the parameter of the algorithm):

Definition 1. *Let $c(1), c(2), \dots$ be the smallest coefficients in $[1, \infty) \cup \{\infty\}$ such that for all i and all sets of policies $\pi_1, \pi_2, \dots, \pi_i$, $\mu P_{\pi_1} P_{\pi_2} \dots P_{\pi_i} \leq c(i)\nu$, and let $C^{(1)}$ and $C^{(2)}$ be the following coefficients in $[1, \infty) \cup \{\infty\}$*

$$C^{(1)} = (1-\gamma)\sum_{i=0}^{\infty} \gamma^i c(i), \qquad C^{(2)} = (1-\gamma)^2 \sum_{i=0}^{\infty}\sum_{j=i}^{\infty} \gamma^j c(j) = (1-\gamma)^2 \sum_{i=0}^{\infty} (i+1)\gamma^i c(i).$$

Remark 1. *A bound involving $C^{(1)}$ is in general better than one involving $C^{(2)}$, in the sense that we always have (i) $C^{(1)} \leq \frac{1}{1-\gamma} C^{(2)} = O(C^{(2)})$ while (ii) we may have $C^{(1)} < \infty$ and $C^{(2)} = \infty$.*
(i) holds because, using the fact that $c(i) \geq 1$, we have

$$C^{(1)} = (1-\gamma)\sum_{i=0}^{\infty} \gamma^i c(i) \leq (1-\gamma)\sum_{i=0}^{\infty}\sum_{j=0}^{\infty} \gamma^{i+j} c(i) = \frac{1}{1-\gamma} C^{(2)}.$$

Now (ii) can be obtained in any situation where $c(i) = \Theta(\frac{1}{i^2\gamma^i})$, since the generic term of $C^{(1)}$ is $\gamma^i c(i) = \Theta(\frac{1}{i^2})$ (the infinite sum converges) while that of $C^{(2)}$ is $(i+1)\gamma^i c(i) = \Theta(\frac{1}{i})$ (the infinite sum diverges to infinity).

With these coefficients in hand, we can prove the following performance bounds.

Theorem 1. *At each iteration k of DPI (Algorithm 1), the expected loss satisfies:*

$$\mu(v_{\pi_*} - v_{\pi_k}) \leq \frac{C^{(2)}}{(1-\gamma)^2} \max_{1\leq i\leq k} \epsilon_i + \gamma^k V_{\max} \quad \text{and} \quad \mu(v_{\pi_*} - v_{\pi_k}) \leq \frac{C^{(1)}}{1-\gamma} \sum_{i=1}^{k} \epsilon_i + \gamma^k V_{\max}.$$

Proof. A proof of the first bound can be found in Lazaric *et al.* (2010). For completeness, we provide one in Appendix A, along with the proof of the (original) second bound, all the more that they share a significant part of the arguments. □

Though the second bound involves the sum of the errors instead of the max value, as noted in Remark 1 its coefficient $C^{(1)}$ is better than $C^{(2)}$. As stated in the following corollary, the above theorem implies that the asymptotic performance bound is approched after a small number (or order $O(\log \frac{1}{\epsilon})$) of iterations.

Corollary 1. *Write* $\epsilon = \max_{1 \leq i \leq k} \epsilon_i$.

$$\text{If } k \geq \frac{\log \frac{V_{\max}}{\epsilon}}{1-\gamma}, \quad \text{then } \mu(v_{\pi_*} - v_{\pi_k}) \leq \left(\frac{C^{(2)}}{(1-\gamma)^2} + 1 \right) \epsilon.$$

$$\text{If } k = \left\lceil \frac{\log \frac{V_{\max}}{\epsilon}}{1-\gamma} \right\rceil, \quad \text{then } \mu(v_{\pi_*} - v_{\pi_k}) \leq \left(\frac{kC^{(1)}}{1-\gamma} + 1 \right) \epsilon \leq \left(\frac{\left(\log \frac{V_{\max}}{\epsilon} + 1 \right) C^{(1)}}{(1-\gamma)^2} + 1 \right) \epsilon.$$

4 Conservative Policy Iteration (CPI)

We now turn to the description of Conservative Policy Iteration (CPI) proposed by Kakade and Langford (2002). At iteration k, CPI (described in Algorithm 2) uses the distribution $d_{\pi_k,\nu} = (1-\gamma)\nu(I - \gamma P_{\pi_k})^{-1}$—the discounted occupancy measure induced by π_k when starting from μ—for calling the approximate greedy operator and for deciding whether to stop or not. Furthermore, it uses an adaptive stepsize α to generate a stochastic mixture of all the policies that are returned by the successive calls to the approximate greedy operator, which explains the adjective "conservative".

Algorithm 2 CPI

input: a distribution ν, an initial policy π_0, $\rho > 0$
for $k = 0, 1, \ldots$ **do**
  $\pi'_{k+1} \leftarrow \mathcal{G}_{\epsilon_{k+1}}(d_{\pi_k,\nu}, v_{\pi_k})$
  Compute a $\frac{\rho}{3}$-accurate estimate $\hat{\mathbb{A}}_{k+1}$ of $\mathbb{A}_{k+1} = d_{\pi_k,\nu}(T_{\pi'_{k+1}} v_{\pi_k} - v_{\pi_k})$
  if $\hat{\mathbb{A}}_{k+1} \leq \frac{2\rho}{3}$ **then**
    return: π_k
  end if
  $\alpha_{k+1} \leftarrow \frac{(1-\gamma)(\hat{\mathbb{A}}_{k+1} - \frac{\rho}{3})}{4\gamma V_{\max}}$
  $\pi_{k+1} \leftarrow (1 - \alpha_{k+1})\pi_k + \alpha_{k+1}\pi'_{k+1}$
end for

The analysis here relies on the following coefficient:

Definition 2. *Let* C_{π_*} *be the smallest coefficient in* $[1, \infty) \cup \{\infty\}$ *such that* $d_{\pi_*,\mu} \leq C_{\pi_*}\nu$.

Remark 2. *Our motivation for revisiting CPI is related to the fact that the constant* C_{π_*} *that will appear soon in its analysis is better than* $C^{(1)}$ *(and thus also, by Remark 1, better that* $C^{(2)}$*) of algorithms like DPI*[1] *in the sense that (i) we always have* $C_{\pi_*} \leq C^{(1)}$ *and (ii) we may have* $C_{\pi_*} < \infty$ *and* $C^{(1)} = \infty$*; moreover, if for any MDP and distribution* μ*, there always exists a parameter* ν *such that* $C_{\pi_*} < \infty$*, there might not exist a* ν *such that* $C^{(2)} < \infty$.
(i) holds because

$$d_{\pi_*,\mu} = (1-\gamma)\mu(I - \gamma P_{\pi_*})^{-1} = (1-\gamma)\sum_{i=0}^{\infty} \gamma^i \mu (P_{\pi_*})^i \leq (1-\gamma)\sum_{i=0}^{\infty} \gamma^i c(i)\nu = C^{(1)}\nu$$

and C_{π_*} *is the smallest coefficient satisfying the above inequality.*
Now consider (ii). The positive part of the claim ("always exists") is trivial: it is sufficient to take $\nu = d_{\pi_*,\mu}$ *and we have* $C_{\pi_*} = 1$*. The negative part ("there might not exist") can be shown by considering an MDP defined on* $\mathbb{N}$*,* μ *equal to the dirac measure* $\delta(\{0\})$ *on state* 0 *and an infinite number of actions* $a \in \mathbb{N}$ *that result in a deterministic transition from* 0 *to* a*. As in Definition 1, let* $c(1) \in [1, \infty) \cup \{\infty\}$ *be such that for all* π*,* $\mu P_\pi \leq c(1)\nu$*. Then for all actions* a*, we have* $\delta(\{a\}) \leq c(1)\nu$*. As a consequence,* $1 = \sum_{i \in \mathbb{N}} \nu(i) \geq \frac{1}{c(1)} \sum_{i \in N} 1$ *and thus necessarily* $c(1) = \infty$*. As a consequence* $C^{(2)} = \infty$.

The constant C_{π_*} will be small when the parameter distribution ν is chosen so that it fits as much as possible $d_{\pi_*,\mu}$ that is the discounted expected long-term occupancy of the optimal policy π_* starting from

[1] Though we do not develop this here, it can be seen that concentrability coefficients that have been introduced for other approximate Dynamic Programming algorithms like Value Iteration (see Munos and Szepesvári (2008); Farahmand *et al.* (2010)) or Modified Policy Iteration (see Scherrer *et al.* (2012)) are equal to $C^{(2)}$.

μ. A good domain expert should be able to provide such an estimate, and thus the condition $C_{\pi_*} < \infty$ is rather mild. We have the following performance bound[2].

Theorem 2. *CPI (Algorithm 2) has the following properties:*
(i) The expected values $\nu v_{\pi_k} = \mathbb{E}[v_{\pi_k}(s)|s \sim \nu]$ of policies π_k starting from distribution ν are monotonically increasing: $\nu v_{k+1} > \nu v_k + \frac{\rho^2}{72\gamma {V_{\max}}^2}$.
(ii) The (random) iteration k^* at which the algorithm stops is such that $k^* \leq \frac{72\gamma {V_{\max}}^2}{\rho^2}$.
(iii) The policy π_{k^*} that is returned satisfies

$$\mu(v_{\pi_*} - v_{\pi_{k^*}}) \leq \frac{C_{\pi_*}}{(1-\gamma)^2}(\epsilon_{k^*+1} + \rho).$$

Proof. The proof follows the lines of that Kakade and Langford (2002) and is provided in Appendix B for completeness. □

We have the following immediate corollary that shows that CPI obtains a performance bounds similar to those of DPI (in Corollary 1)—at the exception that it involves a different (better) concentrability constant—after $O(\frac{1}{\epsilon^2})$ iterations.

Corollary 2. *If CPI is run with parameter $\rho = \epsilon = \max_{1 \leq i \leq k} \epsilon_i$, then CPI stops after at most $\frac{72\gamma {V_{\max}}^2}{\epsilon^2}$ iterations and returns a policy π_{k^*} that satisfies:*

$$\mu(v_{\pi_*} - v_{\pi_{k^*}}) \leq \frac{2C_{\pi_*}}{(1-\gamma)^2}\epsilon.$$

We also provide a complementary original analysis of this algorithm that further highlights its connection with DPI.

Theorem 3. *At each iteration $k < k^*$ of CPI (Algorithm 2), the expected loss satisfies:*

$$\mu(v_{\pi_*} - v_{\pi_k}) \leq \frac{C^{(1)}}{(1-\gamma)^2} \sum_{i=1}^{k} \alpha_i \epsilon_i + e^{\left\{(1-\gamma)\sum_{i=1}^{k} \alpha_i\right\}} V_{\max}.$$

Proof. The proof is a natural but tedious extension of the analysis of DPI to the situation where conservative steps are made, and is deferred to Appendix C. □

Since in the proof of Theorem 2 (in Appendix B), one shows that the learning steps of CPI satisfy $\alpha_k \geq \frac{(1-\gamma)\rho}{12\gamma V_{\max}}$, the right term $e^{\left\{(1-\gamma)\sum_{i=1}^{k} \alpha_i\right\}}$ above tends 0 exponentially fast, and we get the following corollary that shows that CPI has a performance bound with the coefficient $C^{(1)}$ of DPI in a number of iterations $O(\frac{\log \frac{1}{\epsilon}}{\epsilon})$.

Corollary 3. *Assume CPI is run with parameter $\rho = \epsilon = \max_{1 \leq i \leq k} \epsilon_i$. The smallest (random) iteration $k^\dagger$ such that $\frac{\log \frac{V_{\max}}{\epsilon}}{1-\gamma} \leq \sum_{i=1}^{k^\dagger} \alpha_i \leq \frac{\log \frac{V_{\max}}{\epsilon}}{1-\gamma} + 1$ is such that $k^\dagger \leq \frac{12\gamma V_{\max} \log \frac{V_{\max}}{\epsilon}}{\epsilon(1-\gamma)^2}$ and the policy $\pi_{k^\dagger}$ satisfies:*

$$\mu(v_{\pi_*} - v_{\pi_{k^\dagger}}) \leq \left(\frac{C^{(1)}\left(\sum_{i=1}^{k^\dagger} \alpha_i\right)}{(1-\gamma)^2} + 1\right)\epsilon \leq \left(\frac{C^{(1)}\left(\log \frac{V_{\max}}{\epsilon} + 1\right)}{(1-\gamma)^3} + 1\right)\epsilon.$$

In practice, the choice for the learning step α_k in Algorithm 2 is very conservative, which makes CPI (as it is) a very slow algorithm. Natural solutions to this problem, that have for instance been considered in a variation of CPI for search-based structure prediction problems (III *et al.*, 2009; Daumé III *et al.*, 2006), is to either use a line-search (to optimize the learning step $\alpha_k \in (0, 1)$) or even to use a fixed value

[2]Note that there are two small differences between the algorithm and analysis described by Kakade and Langford (2002) and the ones we give here: 1) the stepsize α is a factor $\frac{1}{\gamma}$ bigger in our description, and thus the number of iterations is slightly better (smaller by a factor γ); 2) our result is stated in terms of the error ϵ_k that may not be known in advance and the input parameter ρ while Kakade and Langford (2002) assume a uniform bound ϵ on the errors (ϵ_k) is known and equal to the parameter ρ.

α (e.g. $\alpha = 0.1$) for all iterations. This latter solution, that one may call CPI(α), works indeed well in practice, and is significantly simpler to implement since one is relieved of the necessity to estimate $\hat{\mathbb{A}}_{k+1}$ through rollouts (see Kakade and Langford (2002) for the description of this process); indeed it becomes almost as simple as DPI except that one uses the distribution $d_{\pi_k,\nu}$ and conservative steps. Since the proof is based on a generalization of the analysis of DPI and thus does not use any of the specific properties of CPI, it turns out that the results we have just given (Corollary 3) can straightforwardly be specialized to the case of this algorithm.

Corollary 4. *Assume we run CPI(α) for some $\alpha \in (0,1)$, that is CPI (Algorithm 2) with $\alpha_k = \alpha$ for all k. Write $\epsilon = \max_{1 \le i \le k} \epsilon_i$.*

$$\text{If } k = \left\lceil \frac{\log \frac{V_{\max}}{\epsilon}}{\alpha(1-\gamma)} \right\rceil, \quad \text{then } \mu(v_{\pi_*} - v_{\pi_k}) \le \frac{\alpha(k+1)C^{(1)}}{(1-\gamma)^2}\epsilon \le \left(\frac{C^{(1)}\left(\log \frac{V_{\max}}{\epsilon} + 1\right)}{(1-\gamma)^3} + 1 \right) \epsilon.$$

We see here that the parameter α directly controls the rate of CPI(α)[3]. Furthermore, if one sets α to a sufficiently small value, one should recover the nice properties of Theorem 2-Corollary 2: monotonicity of the expected value and a performance bound with respect to the best constant C_{π_*}, though we do not formalize this here.

In summary, Corollary 2 and Remark 2 tell us that CPI has a performance guarantee that can be arbitrarily better than that of DPI, though the opposite is not true. This, however, comes at the cost of a significant exponential increase of time complexity since Corollary 2 states that there might be a number of iterations that scales in $O(\frac{1}{\epsilon^2})$, while the guarantee of DPI (Corollary 1) only requires $O\left(\log \frac{1}{\epsilon}\right)$ iterations. When the analysis of CPI is relaxed so that the performance guarantee is expressed in terms of the (worse) coefficient $C^{(1)}$ of DPI (Corollary 3), we were able to slightly improve the rate—by a factor $\tilde{O}(\epsilon)$—, though it is still exponentially slower than that of DPI. The algorithm that we present in the next section will be the best of both worlds: it will enjoy a performance guarantee involving the best constant C_{π_*}, but with a time complexity similar to that of DPI.

5 Non-stationary Direct Policy Iteration (NSDPI)

We are now going to describe an algorithm that has a flavour similar to DPI – in the sense that at each step it does a full step towards a new policy – but also has a conservative flavour like CPI – in the sense that the policies will evolve more and more slowly. This algorithm is based on finite-horizon non-stationary policies. We will write $\sigma = \pi_1 \pi_2 \dots \pi_k$ the k-horizon policy that makes the first action according to π_1, then the second action according to π_2, etc. Its value is $v_\sigma = T_{\pi_1} T_{\pi_2} \dots T_{\pi_k} r$. We will write $\sigma = \varnothing$ the "empty" non-stationary policy. Note that $v_\varnothing = r$ and that any infinite-horizon policy that begins with $\sigma = \pi_1 \pi_2 \dots \pi_k$, which we will denote "$\sigma \dots$" has a value $v_{\sigma\dots} \ge v_\sigma - \gamma^k V_{\max}$.

The last algorithm we consider, named here Non-Stationary Direct Policy Iteration (NSDPI) because it behaves as DPI but builds a non-stationary policy by iteratively concatenating the policies that are returned by the approximate greedy operator, is described in Algorithm 3.

Algorithm 3 NSDPI

input: a distribution ν
initialization: $\sigma_0 = \varnothing$
for $k = 0, 1, 2, \dots$ **do**
  $\pi_{k+1} \leftarrow \mathcal{G}_{\epsilon_{k+1}}(\nu, v_{\sigma_k})$
  $\sigma_{k+1} \leftarrow \pi_{k+1}\sigma_k$
end for

We are going to state a performance bound for this algorithm with respect to the constant C_{π_*}, but also an alternative bound based on the following new concentrability coefficients.

[3]The performance bound of CPI(1) (with $\alpha = 1$) does not match the bound of DPI (Corollary 1), but is a factor $\frac{1}{1-\gamma}$ worse. This amplification is due to the fact that the approximate greedy operator uses the distribution $d_{\pi_k,\nu} \ge (1-\gamma)\nu$ instead of ν (for DPI).

Definition 3. *Let $c_{\pi_*}(1), c_{\pi_*}(2), \dots$ be the smallest coefficients in $[1,\infty) \cup \{\infty\}$ such that for all i, $\mu(P_{\pi_*})^i \le c_{\pi_*}(i)\nu$. and let $C^{(1)}_{\pi_*}$ be the following coefficient in $[1,\infty) \cup \{\infty\}$:*

$$C^{(1)}_{\pi_*} = (1-\gamma)\sum_{i=0}^{\infty} \gamma^i c_\pi(i).$$

Remark 3. *A bound involving C_{π_*} is in general better than one involving $C^{(1)}_{\pi_*}$ in the sense that (i) we always have $C_{\pi_*} \le C^{(1)}_{\pi_*}$ while (ii) we may have $C_{\pi_*} < \infty$ and $C^{(1)}_{\pi_*} = \infty$. Similarly, a bound involving $C^{(1)}_{\pi_*}$ is in general better than one involving $C^{(1)}$ since (iii) we have $C^{(1)}_{\pi_*} \le C^{(1)}$ while (iv) we may have $C^{(1)}_{\pi_*} < \infty$ and $C^{(1)} = \infty$.*
(i) holds because (very similarly to Remark 2-(i))

$$d_{\pi_*,\mu} = (1-\gamma)\mu(I-\gamma P_{\pi_*})^{-1} = (1-\gamma)\sum_{i=0}^{\infty}\gamma^i \mu(P_{\pi_*})^i \le (1-\gamma)\sum_{i=0}^{\infty}\gamma^i c_{\pi_*}(i)\nu = C^{(1)}_{\pi_*}\nu$$

and C_{π_*} is the smallest coefficient satisfying the above inequality.
(ii) can easily be obtained by designing a problem with $c_{\pi_*}(1) = \infty$ as in Remark 2-(ii).
(iii) is a consequence of the fact that for all i, $c_{\pi_*}(i) \le c(i)$.
Finally, (iv) is trivially obtained by considering two different policies.

With this notations in hand, we are ready to state that NSDPI (Algorithm 3) enjoys two guarantees that have a fast rate like those of DPI (Theorem 1), one expressed in terms of the concentrability C_{π_*} that was introduced for CPI (in Definition 2), and the other in terms of the constant $C^{(1)}_{\pi_*}$ we have just introduced.

Theorem 4. *At each iteration k of NSDPI (Algorithm 3), the expected loss of running an infinitely-long policy that begins by σ_k satisfies*

$$\mu(v_{\pi_*} - v_{\sigma_k \dots}) \le \frac{C^{(1)}_{\pi_*}}{1-\gamma}\max_{1\le i\le k}\epsilon_i + 2\gamma^k V_{\max} \quad \text{and} \quad \mu(v_{\pi_*} - v_{\sigma_k \dots}) \le \frac{C_{\pi_*}}{1-\gamma}\sum_{i=1}^{k}\epsilon_i + 2\gamma^k V_{\max}.$$

Proof. The proof of Theorem 4 is rather simple (much simpler than the previous ones), and is deferred to Appendix D. □

As shown in the following immediate corollary of Theorem 4, these relations constitute guarantees that small errors ϵ_i in the greedy operator *quickly* induce good policies.

Corollary 5. *Write $\epsilon = \max_{1\le i\le k}\epsilon_i$.*

$$\text{If } k \ge \frac{\log\frac{2V_{\max}}{\epsilon}}{1-\gamma} \text{ then } \mu(v_{\pi_*} - v_{\sigma_k}) \le \left(\frac{C^{(1)}_{\pi_*}}{1-\gamma} + 1\right)\epsilon.$$

$$\text{If } k = \left\lceil \frac{\log\frac{2V_{\max}}{\epsilon}}{1-\gamma} \right\rceil \text{ then } \mu(v_{\pi_*} - v_{\pi_k}) \le \left(\frac{kC_{\pi_*}}{1-\gamma} + 1\right)\epsilon \le \left(\frac{\left(\log\frac{2V_{\max}}{\epsilon} + 1\right)C_{\pi_*}}{(1-\gamma)^2} + 1\right)\epsilon.$$

The first bound has a better dependency with respect to $\frac{1}{1-\gamma}$, but (as explained in Remark 3) is expressed in terms of the worse coefficient $C^{(1)}_{\pi_*}$. The second guarantee is almost as good as that of CPI (Corollary 2) since it only contains an extra $\log\frac{1}{\epsilon}$ term, but it has the nice property that it holds quickly: in time $\log\frac{1}{\epsilon}$ instead of $O(\frac{1}{\epsilon^2})$, that is exponentially faster.

We devised NSDPI as a DPI-like simplified variation of one of the non-stationary dynamic programming algorithm recently introduced by Scherrer and Lesner (2012), the *Non-Stationary PI algorithm with growing period*. The main difference with NSDPI is that one considers there the value $v_{(\sigma_k)^\infty}$ of the policy that loops infinitely on σ_k instead of the value v_{σ_k} of the only first k steps here. Following the intuition that when k is big, these two values will be close to each other, we ended up focusing on NSDPI because it is simpler. Remarkably, NSDPI turns out to be almost identical to an older algorithm, the *Policy Search by Dynamic Programming* (PSDP) algorithm Bagnell *et al.* (2003). It should

	Performance Bound			Nb. Iterations	Reference in text
DPI (Alg. 1)	$C^{(2)}$	$\frac{1}{(1-\gamma)^2}$	ϵ	$\log \frac{1}{\epsilon}$	Corollary 1
	$C^{(1)}$	$\frac{1}{(1-\gamma)^2}$	$\epsilon \log \frac{1}{\epsilon}$	$\log \frac{1}{\epsilon}$	
CPI(α)	$C^{(1)}$	$\frac{1}{(1-\gamma)^2}$	ϵ	$\frac{1}{\alpha} \log \frac{1}{\epsilon}$	Corollary 4
CPI (Alg. 2)	$C^{(1)}$	$\frac{1}{(1-\gamma)^3}$	$\epsilon \log \frac{1}{\epsilon}$	$\frac{1}{\epsilon} \log \frac{1}{\epsilon}$	Corollary 3
	C_{π_*}	$\frac{1}{(1-\gamma)^2}$	ϵ	$\frac{1}{\epsilon^2}$	Theorem 2
NSDPI (Alg. 3)	C_{π_*}	$\frac{1}{(1-\gamma)^2}$	$\epsilon \log \frac{1}{\epsilon}$	$\log \frac{1}{\epsilon}$	Corollary 5
	$C^{(1)}_{\pi_*}$	$\frac{1}{1-\gamma}$	ϵ	$\log \frac{1}{\epsilon}$	

Figure 1: Comparison of the algorithms.

however be noted that PSDP was introduced for a slightly different control problem where the horizon is finite, while we are considering here the infinite-horizon problem. Given a problem with horizon T, PSDP comes with a guarantee that is essentially identical to the first bound in Corollary 5, but requiring as many input distributions as there are time steps[4]. Our main contribution with respect to PSDP is that by considering the infinite-horizon case, we managed to require only one input parameter (the distribution ν that should match as much as possible the measure $d_{\pi_*,\mu}$, recall Definition 1)—a much milder assumption—and provide the second performance bound with respect to C_{π_*}, that is better (cf. Remark 3).

6 Discussion, Conclusion and Future Work

In this article, we have described two algorithms of the literature, DPI and CPI, and introduced the NSDPI algorithm that borrows ideas from both DPI and CPI, while also having some very close algorithmic connections with PSDP Bagnell *et al.* (2003) and the *Non-Stationary PI algorithm with growing period* of Scherrer and Lesner (2012). Figure 1 synthesizes the theoretical guarantees we have discussed about these algorithms. For each such guarantee, we provide the dependency of the performance bound and the number of iterations with respect to ϵ, $\frac{1}{1-\gamma}$ and the concentrability coefficients (for CPI, we assume as Kakade and Langford (2002) that $\rho = \epsilon$). We highlight in red the bounds that are to our knowledge new[5].

One of the most important message of our work is that what is usually hidden in the constants of the performance bounds does matter. The constants we discussed can be sorted from the worst to the best (cf. Remarks 1, 2 and 3) as follows: $C^{(2)}, C^{(1)}, C^{(1)}_{\pi_*}, C_{\pi_*}$. To our knowledge, this is the first time that such an in-depth comparison of the bounds is done, and our hierarchy of constants has interesting implications that go beyond to the policy search algorithms we have been focusing on in this paper. As a matter of fact, several dynamic programming algorithms—AVI (Munos, 2007), API (Munos, 2003), λPI Scherrer (2013), AMPI (Scherrer *et al.*, 2012)—come with guarantees involving the worst constant $C^{(2)}$, that can easily be made infinite. On the positive side, we have argued that the guarantee for CPI can be arbitrarily stronger than the other state-of-the-art algorithms. We have identified the NSDPI algorithm as having a similar nice property. Furthermore, we can observe that DPI and NSDPI both have the best time complexity guarantee. Thus, NSDPI turns out to have the overall best guarantees.

At the technical level, several of our bounds come in pair, and this is due to the fact that we have introduced a new proof technique in order to derive new bounds with various constants. This led to a new bound for DPI, that is better since it involves the $C^{(1)}$ constant instead of $C^{(2)}$. It also enabled us to derive new bounds for CPI (and its natural algorithmic variant CPI(α)) that is worse in terms of

[4] It is assumed by Bagnell *et al.* (2003) that a set of baseline distributions $\nu_1, \nu_2, \ldots, \nu_T$ provide estimates of where the optimal policy is leading the system from some distribution μ *at all* time steps $1, 2, \ldots, T$; then, the authors measure the mismatch through coefficients $c_{\pi_*}(i)$ such that $\mu P^i_{\pi_*} \leq c_{\pi_*}(i)\nu_i$. This kind of assumption is essentially identical to the concentrability assumption underlying the constant $C^{(1)}_{\pi_*}$ in Definition 3, the only difference being that we only refer to one measure ν.

[5] We do not highlight the second bound of NSDPI, acknowledging that it already appears in a very close form in Bagnell *et al.* (2003) for PSDP.

guarantee but has a better time complexity ($\tilde{O}(\frac{1}{\epsilon})$ instead of $O(\frac{1}{\epsilon})$). We believe this new technique may be helpful in the future for the analysis of other algorithms.

The main limitation of our analysis lies in the assumption, considered all along the paper, that all algorithms have at disposal an ϵ-approximate greedy operator. This is in general not realistic, since it may be hard to control directly the quality level ϵ. Furthermore, it may be unreasonable to compare all algorithms on this basis, since the underlying optimization problems may have slightly different complexities: for instance, methods like CPI look in a space of stochastic policies while DPI moves in a space of deterministic policies. Digging and understanding in more depth what is potentially hidden in the term ϵ—as we have done here for the concentrability constants—constitutes a very natural research direction.

Last but not least, on the practical side, we have run preliminary numerical experiments that somewhat support our theoretical argument that algorithms with a better concentrability constant should be preferred. On simulations on relatively small problems, CPI+ (CPI with a crude line-search mechanism), CPI(α) and NSDPI were shown to always perform significantly better than DPI, NSDPI always displayed the least variability, and CPI(α) performed the best on average. We refer the reader to Appendix E for further details. Running and analyzing similar experiments on bigger domains constitutes interesting future work.

References

Antos, A., Szepesvári, C., and Munos, R. (2008). Learning near-optimal policies with Bellman-residual minimization based fitted policy iteration and a single sample path. *Machine Learning*, **71**(1), 89–129.

Archibald, T., McKinnon, K., and Thomas, L. (1995). On the Generation of Markov Decision Processes. *Journal of the Operational Research Society*, **46**, 354–361.

Bagnell, J., Kakade, S., Ng, A., and Schneider, J. (2003). Policy search by dynamic programming. In *Neural Information Processing Systems*, volume 16. MIT Press.

Bertsekas, D. and Tsitsiklis, J. (1996). *Neuro-Dynamic Programming*. Athena Scientific.

Daumé III, H., Langford, J., and Marcu, D. (2006). Searn in practice.

Farahmand, A., Ghavamzadeh, M., Szepesvári, C., and Mannor, S. (2009). Regularized policy iteration. *Advances in Neural Information Processing Systems*, **21**, 441–448.

Farahmand, A., Munos, R., and Szepesvári, C. (2010). Error propagation for approximate policy and value iteration (extended version). In *NIPS*.

Fern, A., Yoon, S., and Givan, R. (2006). Approximate Policy Iteration with a Policy Language Bias: Solving Relational Markov Decision Processes. *Journal of Artificial Intelligence Research*, **25**, 75–118.

III, H. D., Langford, J., and Marcu, D. (2009). Search-based structured prediction. *Machine Learning*, **75**(3), 297–325.

Kakade, S. (2003). *On the Sample Complexity of Reinforcement Learning*. Ph.D. thesis, University College London.

Kakade, S. and Langford, J. (2002). Approximately optimal approximate reinforcement learning. In *ICML*, pages 267–274.

Lagoudakis, M. and Parr, R. (2003). Reinforcement Learning as Classification: Leveraging Modern Classifiers. In *Proceedings of ICML*, pages 424–431.

Lazaric, A., Ghavamzadeh, M., and Munos, R. (2010). Analysis of a Classification-based Policy Iteration Algorithm. In *Proceedings of ICML*, pages 607–614.

Lazaric, A., Ghavamzadeh, M., and Munos, R. (2011). Finite-Sample Analysis of Least-Squares Policy Iteration. *To appear in Journal of Machine learning Research (JMLR)*.

Munos, R. (2003). Error Bounds for Approximate Policy Iteration. In *International Conference on Machine Learning (ICML)*, pages 560–567.

Munos, R. (2007). Performance Bounds in Lp norm for Approximate Value Iteration. *SIAM J. Control and Optimization*.

Munos, R. and Szepesvári, C. (2008). Finite time bounds for sampling based fitted value iteration. *Journal of Machine Learning Research (JMLR)*, **9**, 815–857.

Puterman, M. (1994). *Markov Decision Processes*. Wiley, New York.

Scherrer, B. (2013). Performance Bounds for Lambda Policy Iteration and Application to the Game of Tetris. *Journal of Machine Learning Research*, **14**, 1175–1221.

Scherrer, B. and Lesner, B. (2012). On the Use of Non-Stationary Policies for Stationary Infinite-Horizon Markov Decision Processes. In *NIPS 2012 - Neural Information Processing Systems*, South Lake Tahoe, United States.

Scherrer, B., Gabillon, V., Ghavamzadeh, M., and Geist, M. (2012). Approximate Modified Policy Iteration. Research report, INRIA.

Szepesvári, C. (2010). Reinforcement learning algorithms for MDPs. In *Wiley Encyclopedia of Operations Research*. Wiley.

A Proof of Theorem 1

Our proof is here even slightly more general than what is required: we provide the result for any reference policy π (and not only for the optimal policy π_*). Writing $e_{k+1} = \max_{\pi'} T_{\pi'} v_{\pi_k} - T_{\pi_{k+1}} v_{\pi_k}$, we can first see that:

$$\begin{aligned} v_\pi - v_{\pi_{k+1}} &= T_\pi v_\pi - T_\pi v_{\pi_k} + T_\pi v_{\pi_k} - T_{\pi_{k+1}} v_{\pi_k} + T_{\pi_{k+1}} v_{\pi_k} - T_{\pi_{k+1}} v_{\pi_{k+1}} \\ &\leq \gamma P_\pi (v_\pi - v_{\pi_k}) + e_{k+1} + \gamma P_{\pi_{k+1}} (v_{\pi_k} - v_{\pi_{k+1}}). \end{aligned} \tag{2}$$

Using the fact that $v_{\pi_{k+1}} = (I - \gamma P_{\pi_{k+1}})^{-1} r$, one can notice that:

$$\begin{aligned} v_{\pi_k} - v_{\pi_{k+1}} &= (I - \gamma P_{\pi_{k+1}})^{-1} (v_{\pi_k} - \gamma P_{\pi_{k+1}} v_{\pi_k} - r) \\ &= (I - \gamma P_{\pi_{k+1}})^{-1} (T_{\pi_k} v_{\pi_k} - T_{\pi_{k+1}} v_{\pi_k}) \\ &\leq (I - \gamma P_{\pi_{k+1}})^{-1} e_{k+1}. \end{aligned}$$

Putting this back in Equation (2) we get:

$$v_\pi - v_{\pi_{k+1}} = \gamma P_\pi (v_\pi - v_{\pi_k}) + (I - \gamma P_{\pi_{k+1}})^{-1} e_{k+1}.$$

By induction on k we obtain:

$$v_\pi - v_{\pi_k} = \sum_{i=0}^{k-1} (\gamma P_\pi)^i (I - \gamma P_{\pi_{k-i}})^{-1} e_{k-i} + (\gamma P_\pi)^k (v_\pi - v_{\pi_0}).$$

Multiplying both sides by μ (and observing that $e_k \geq 0$) and using Definition 1 and the fact that $\nu e_j \leq \epsilon_j$, we obtain:

$$\begin{aligned} \mu(v_\pi - v_{\pi_k}) &\leq \sum_{i=0}^{k-1} \mu (\gamma P_\pi)^i (I - \gamma P_{\pi_{k-i}})^{-1} e_{k-i} + \gamma^k V_{\max} \\ &\leq \sum_{i=0}^{k-1} \left(\sum_{j=0}^{\infty} \gamma^{i+j} c(i+j) \epsilon_{k-i} \right) + \gamma^k V_{\max} \\ &= \sum_{i=0}^{k-1} \sum_{j=i}^{\infty} \gamma^j c(j) \epsilon_{k-i} + \gamma^k V_{\max} \\ &\leq \sum_{i=0}^{k-1} \sum_{j=i}^{\infty} \gamma^j c(j) \max_{1 \leq l \leq k} \epsilon_l + \gamma^k V_{\max}, \end{aligned} \tag{3}$$

which provides the first bound. Starting back on Equation (3), we get

$$\begin{aligned} \mu(v_\pi - v_{\pi_k}) &\leq \sum_{i=0}^{k-1} \sum_{j=i}^{\infty} \gamma^j c(j) \epsilon_{k-i} + \gamma^k V_{\max} \\ &\leq \sum_{i=0}^{k-1} \sum_{j=0}^{\infty} \gamma^j c(j) \epsilon_{k-i} + \gamma^k V_{\max} \\ &= \sum_{j=0}^{\infty} \gamma^j c(j) \sum_{i=1}^{k} \epsilon_i + \gamma^k V_{\max}, \end{aligned}$$

which proves the second bound.

B Proof of Theorem 2

We include a concise and self-contained proof that essentially follows the steps in (Kakade and Langford, 2002).

(i) We first show that the value $\eta_k = \nu v_{\pi_k}$ is increasing along the iterations. Consider some iteration k of the algorithm. Using the facts that $T_{\pi_{k+1}} v_{\pi_k} = (1-\alpha_{k+1}) v_{\pi_k} + \alpha_{k+1} T_{\pi'_{k+1}} v_{\pi_k}$ and $P_{\pi_{k+1}} = (1-\alpha_{k+1}) P_{\pi_k} + \alpha_{k+1} P_{\pi'_{k+1}}$, we can see that:

$$\begin{aligned}
\eta_{k+1} - \eta_k &= \nu(v_{\pi_{k+1}} - v_{\pi_k}) \\
&= \nu[(I - \gamma P_{\pi_{k+1}})^{-1} r - v_{\pi_k}] \\
&= \nu(I - \gamma P_{\pi_k})^{-1}(I - \gamma P_{\pi_k})(I - \gamma P_{\pi_{k+1}})^{-1}[r - (I - \gamma P_{\pi_{k+1}}) v_{\pi_k}] \\
&= \nu(I - \gamma P_{\pi_k})^{-1}(I - \gamma P_{\pi_{k+1}} + \gamma P_{\pi_{k+1}} - \gamma P_{\pi_k})(I - \gamma P_{\pi_{k+1}})^{-1}(T_{\pi_{k+1}} v_{\pi_k} - v_{\pi_k}) \\
&= \frac{1}{1-\gamma} d_{\pi_k,\nu} \left[I + \gamma \alpha_{k+1} (P_{\pi'_{k+1}} - P_{\pi_k})(I - \gamma P_{\pi_{k+1}})^{-1} \right] \alpha_{k+1} (T_{\pi'_{k+1}} v_{\pi_k} - v_{\pi_k}) \\
&= \alpha_{k+1} \frac{1}{1-\gamma} d_{\pi_k,\nu} (T_{\pi'_{k+1}} v_{\pi_k} - v_{\pi_k}) + \alpha_{k+1}^2 \frac{\gamma}{1-\gamma} d_{\pi_k,\nu} (P_{\pi'_{k+1}} - P_{\pi_k})(I - \gamma P_{\pi_{k+1}})^{-1} (T_{\pi'_{k+1}} v_{\pi_k} - v_{\pi_k}) \\
&\geq \alpha_{k+1} \frac{1}{1-\gamma} (\hat{\mathbb{A}}_{k+1} - \frac{\rho}{3}) - 2\alpha_{k+1}^2 \frac{\gamma}{(1-\gamma)^2} V_{\max}
\end{aligned}$$

where we eventually used the fact that $T_{\pi'_{k+1}} v_{\pi_k} - v_{\pi_k} \in (-V_{\max}, V_{\max})$. Now, it can be seen that the setting $\alpha_{k+1} = \frac{(1-\gamma)(\hat{\mathbb{A}}_{k+1} - \frac{\rho}{3})}{4\gamma V_{\max}}$ of Algorithm 2 is the one that maximizes the above right hand side. With this setting we get:

$$\begin{aligned}
\eta_{k+1} - \eta_k &\geq \frac{(\hat{\mathbb{A}}_{k+1} - \frac{\rho}{3})^2}{8\gamma V_{\max}} \\
&> \frac{\rho^2}{72\gamma V_{\max}}
\end{aligned}$$

since as long as the algorithm iterates, $\hat{\mathbb{A}}_{k+1} > \frac{2\rho}{3}$.
(ii) The second point is a very simple consequence of (i): since $\eta_k = \nu v_{\pi_k} \leq V_{\max}$, the number of iterations of the algorithm cannot exceed $\frac{72\gamma {V_{\max}}^2}{\rho^2}$.
(iii) We now prove the performance bound. Write $e = \max_{\pi'} T_{\pi'} v_{\pi_{k^*}} - v_{\pi_{k^*}}$. We have:

$$\begin{aligned}
v_{\pi^*} - v_{\pi_{k^*}} &= T_{\pi^*} v_{\pi^*} - T_{\pi^*} v_{\pi_{k^*}} + T_{\pi^*} v_{\pi_{k^*}} - v_{\pi_{k^*}} \\
&\leq \gamma P_{\pi^*} (v_{\pi^*} - v_{\pi_{k^*}}) + e \\
&\leq (I - \gamma P_{\pi^*})^{-1} e.
\end{aligned}$$

Multiplying by μ, using Definition 2 and the facts that $e = \max_{\pi'} T_{\pi'} v_{\pi_{k^*}} - T_{\pi_{k^*}} v_{\pi_{k^*}} \geq 0$ and $d_{\pi_{k^*},\nu} \geq (1-\gamma)\nu$, we obtain

$$\begin{aligned}
\mu(v_{\pi^*} - v_{\pi_{k^*}}) &\leq \frac{1}{1-\gamma} d_{\pi^*,\mu} e \\
&\leq \frac{C_{\pi_*}}{1-\gamma} \nu e \\
&\leq \frac{C_{\pi_*}}{(1-\gamma)^2} d_{\pi_{k^*},\nu} e \\
&= \frac{C_{\pi_*}}{(1-\gamma)^2} d_{\pi_{k^*},\nu} (\max_{\pi'} T_{\pi'} v_{\pi_{k^*}} - T_{\pi'_{k^*+1}} v_{\pi_{k^*}} + T_{\pi'_{k^*+1}} v_{\pi_{k^*}} - v_{\pi_{k^*}}) \\
&\leq \frac{C_{\pi_*}}{(1-\gamma)^2} (\epsilon_{k^*} + A_{k+1}).
\end{aligned}$$

The result follows by observing that the advantage satisfies $\mathbb{A}_{k+1} \leq \hat{\mathbb{A}}_{k+1} + \frac{\rho}{3} \leq \rho$ since $\hat{\mathbb{A}}_{k+1} \leq \frac{2\rho}{3}$ when the algorithms stops.

C Proof of Theorem 3

Using the facts that $T_{\pi_{k+1}} v_{\pi_k} = (1-\alpha_{k+1}) v_{\pi_k} + \alpha_{k+1} T_{\pi_{k+1}} v_{\pi_k}$ and the notation $e_{k+1} = \max_{\pi'} T_{\pi'} v_{\pi_k} - T_{\pi'_{k+1}} v_{\pi_k}$, we have:

$$\begin{aligned}
v_\pi - v_{\pi_{k+1}} &= v_\pi - T_{\pi_{k+1}} v_{\pi_k} + T_{\pi_{k+1}} v_{\pi_k} - T_{\pi_{k+1}} v_{\pi_{k+1}} \\
&= v_\pi - (1-\alpha_{k+1}) v_{\pi_k} - \alpha_{k+1} T_{\pi'_{k+1}} v_{\pi_k} + \gamma P_{\pi_{k+1}} (v_{\pi_k} - v_{\pi_{k+1}}) \\
&= (1-\alpha_{k+1})(v_\pi - v_{\pi_k}) + \alpha_{k+1}(T_\pi v_\pi - T_\pi v_{\pi_k}) + \alpha_{k+1}(T_\pi v_{\pi_k} - T_{\pi'_{k+1}} v_{\pi_k}) + \gamma P_{\pi_{k+1}} (v_{\pi_k} - v_{\pi_{k+1}}) \\
&\leq \left[(1-\alpha_{k+1}) I + \alpha_{k+1} \gamma P_\pi\right](v_\pi - v_{\pi_k}) + \alpha_{k+1} e_{k+1} + \gamma P_{\pi_{k+1}} (v_{\pi_k} - v_{\pi_{k+1}}). \qquad (4)
\end{aligned}$$

Using the fact that $v_{\pi_{k+1}} = (I - \gamma P_{\pi_{k+1}})^{-1} r$, we can see that

$$\begin{aligned}
v_{\pi_k} - v_{\pi_{k+1}} &= (I - \gamma P_{\pi_{k+1}})^{-1} (v_{\pi_k} - \gamma P_{\pi_{k+1}} v_{\pi_k} - r) \\
&= (I - \gamma P_{\pi_{k+1}})^{-1} (T_{\pi_k} v_{\pi_k} - T_{\pi_{k+1}} v_{\pi_k}) \\
&\leq (I - \gamma P_{\pi_{k+1}})^{-1} \alpha_{k+1} e_{k+1}.
\end{aligned}$$

Putting this back in Equation (5), we obtain:

$$v_\pi - v_{\pi_{k+1}} \leq \left[(1-\alpha_{k+1}) I + \alpha_{k+1} \gamma P_\pi\right](v_\pi - v_{\pi_k}) + \alpha_{k+1} (I - \gamma P_{\pi_{k+1}})^{-1} e_{k+1}.$$

Define the matrix $Q_k = [(1-\alpha_k) I + \alpha_k \gamma P_\pi]$, the set $\mathcal{N}_{i,k} = \{j; k-i+1 \leq j \leq k\}$ (this set contains exactly i elements), the matrix $R_{i,k} = \prod_{j \in \mathcal{N}_{i,k}} Q_j$, and the coefficients $\beta_k = 1 - \alpha_k(1-\gamma)$ and $\delta_k = \prod_{i=1}^k \beta_k$. We get by induction

$$v_\pi - v_{\pi_k} \leq \sum_{i=0}^{k-1} R_{i,k} \alpha_{k-i} (I - \gamma P_{\pi_{k-i}})^{-1} e_{k-i} + \delta_k V_{\max}. \qquad (5)$$

Let $\mathcal{P}_j(\mathcal{N}_{i,k})$ the set of subsets of $\mathcal{N}_{i,k}$ of size j. With this notation we have

$$R_{i,k} = \sum_{j=0}^{i} \sum_{I \in \mathcal{P}_j(\mathcal{N}_{i,k})} \zeta_{I,i,k} (\gamma P_\pi)^j$$

where for all subset I of $\mathcal{N}_{i,k}$, we wrote

$$\zeta_{I,i,k} = \left(\prod_{n \in I} \alpha_n\right) \left(\prod_{n \in \mathcal{N}_{i,k} \setminus I} (1-\alpha_n)\right).$$

Therefore, by multiplying Equation (5) by μ, using Definition 1, and the facts that $\nu \leq (1-\gamma)d_{nu,\pi_{k+1}}$, we obtain:

$$\begin{aligned}
\mu(v_\pi - v_{\pi_k}) &\leq \frac{1}{1-\gamma}\sum_{i=0}^{k-1}\sum_{j=0}^{i}\sum_{l=0}^{\infty}\sum_{I\in\mathcal{P}_j(\mathcal{N}_{i,k})}\zeta_{I,i,k}\gamma^{j+l}c(j+l)\alpha_{k-i}\epsilon_{k-i} + \delta_k V_{\max}. \\
&= \frac{1}{1-\gamma}\sum_{i=0}^{k-1}\sum_{j=0}^{i}\sum_{l=j}^{\infty}\sum_{I\in\mathcal{P}_j(\mathcal{N}_{i,k})}\zeta_{I,i,k}\gamma^{l}c(l)\alpha_{k-i}\epsilon_{k-i} + \delta_k V_{\max} \\
&\leq \frac{1}{1-\gamma}\sum_{i=0}^{k-1}\sum_{j=0}^{i}\sum_{l=0}^{\infty}\sum_{I\in\mathcal{P}_j(\mathcal{N}_{i,k})}\zeta_{I,i,k}\gamma^{l}c(l)\alpha_{k-i}\epsilon_{k-i} + \delta_k V_{\max} \\
&= \frac{1}{1-\gamma}\left(\sum_{l=0}^{\infty}\gamma^l c(l)\right)\sum_{i=0}^{k-1}\left(\sum_{j=0}^{i}\sum_{I\in\mathcal{P}_j(\mathcal{N}_{i,k})}\zeta_{I,i,k}\right)\alpha_{k-i}\epsilon_{k-i} + \delta_k V_{\max} \\
&= \frac{1}{1-\gamma}\left(\sum_{l=0}^{\infty}\gamma^l c(l)\right)\sum_{i=0}^{k-1}\left(\prod_{j\in\mathcal{N}_{i,k}}(1-\alpha_j+\alpha_j)\right)\alpha_{k-i}\epsilon_{k-i} + \delta_k V_{\max} \\
&= \frac{1}{1-\gamma}\left(\sum_{l=0}^{\infty}\gamma^l c(l)\right)\left(\sum_{i=0}^{k-1}\alpha_{k-i}\epsilon_{k-i}\right) + \delta_k V_{\max}.
\end{aligned}$$

Now, using the fact that for $x \in (0,1)$, $\log(1-x) \leq -x$, we can observe that

$$\begin{aligned}
\log\delta_k &= \log\prod_{i=1}^{k}\beta_i \\
&= \sum_{i=1}^{k}\log\beta_i \\
&= \sum_{i=1}^{k}\log(1-\alpha_i(1-\gamma)) \\
&\leq -(1-\gamma)\sum_{i=1}^{k}\alpha_i
\end{aligned}$$

As a consequence, we get $\delta_k \leq e^{-(1-\gamma)\sum_{i=1}^{k}\alpha_i}$.

D Proof of Theorem 4

We begin by the first relation. For all k, we have

$$v_\pi - v_{\sigma_k} = T_\pi v_\pi - T_\pi v_{\sigma_{k-1}} + T_\pi v_{\sigma_{k-1}} - T_{\pi_k} v_{\sigma_{k-1}} \leq \gamma P_\pi(v_\pi - v_{\sigma_{k-1}}) + e_k$$

where we defined $e_k = \max_{\pi'} T_{\pi'} v_{\sigma_{k-1}} - T_{\pi_k} v_{\sigma_{k-1}}$. By induction, we deduce that

$$v_\pi - v_{\sigma_k} \leq \sum_{i=0}^{k-1}(\gamma P_\pi)^i e_{k-i} + \gamma^k V_{\max}.$$

By multiplying both sides of by μ, using Definition 3 and the fact that $\nu_j e_j \leq \epsilon$, we get:

$$\begin{aligned}\mu(v_\pi - v_{\sigma_k}) &\leq \sum_{i=0}^{k-1} \mu(\gamma P_\pi)^i e_{k-i} + \gamma^k V_{\max} \\ &\leq \sum_{i=0}^{k-1} \gamma^i c(i) \epsilon_{k-i} + \gamma^k V_{\max} \\ &\leq \left(\sum_{i=0}^{k-1} \gamma^i c(i)\right) \max_{1\leq i\leq k} \epsilon_i + \gamma^k V_{\max}.\end{aligned} \tag{6}$$

We now prove the second relation. Starting back in Equation (6) and using Definition 2 (in particular the fact that for all i, $\mu(\gamma P_\pi)^i \leq \frac{1}{1-\gamma} d_{\pi^*,\mu} \leq \frac{C_{\pi_*}}{1-\gamma}\nu$) and the fact that $\nu e_j \leq \epsilon_j$, we get:

$$\mu(v_\pi - v_{\sigma_k}) \leq \sum_{i=0}^{k-1} \mu(\gamma P_\pi)^i e_{k-i} + \gamma^k V_{\max} \leq \frac{C_{\pi_*}}{1-\gamma} \sum_{i=1}^{k} \epsilon_i + \gamma^k V_{\max}$$

and the result is obtained by using the fact that $v_{\sigma_k \ldots} \geq v_{\sigma_k} - \gamma^k V_{\max}$.

E Experiments

In this section, we present some experiments in order to illustrate the different algorithms discussed in the paper and to get some insight on their empirical behaviour. CPI as it is described in Algorithm 2 can be very slow (in one sample experiment on a 100 state problem, it made very slow progress and took several millions of iterations before it stopped) and we did not evaluate it further. Instead, we considered two variations: CPI+ that is identical to CPI except that it chooses the step α at each iteration by doing a line-search towards the policy output by the classifier[6], and CPI(α) with $\alpha = 0.1$, that makes "relatively but not too small" steps at each iteration. We begin by describing the domain and the approximation considered.

Domain and Approximations In order to assess their quality, we consider finite problems where the exact value function can be computed. More precisely, we consider Garnet problems (Archibald *et al.*, 1995), which are a class of randomly constructed finite MDPs. They do not correspond to any specific application, but are totally abstract while remaining representative of the kind of MDP that might be encountered in practice. In our experiments, a Garnet is parameterized by 4 parameters and is written $G(n_S, n_A, b, p)$: n_S is the number of states, n_A is the number of actions, b is a branching factor specifying how many possible next states are possible for each state-action pair (b states are chosen uniformly at random and transition probabilities are set by sampling uniform random $b-1$ cut points between 0 and 1) and p is the number of features (for linear function approximation). The reward is state-dependent: for a given randomly generated Garnet problem, the reward for each state is uniformly sampled between 0 and 1. Features are chosen randomly: Φ is a $n_S \times p$ feature matrix of which each component is randomly and uniformly sampled between 0 and 1. The discount factor γ is set to 0.99 in all experiments.

All the algorithms we have discussed in the paper need to repeatedly compute $\mathcal{G}_\epsilon(\nu, v)$. In other words, they must be able to make calls to an approximate greedy operator applied to the value v of some policy for some distribution ν. To implement this operator, we compute a noisy estimate of the value v with a uniform white noise $u(\iota)$ of amplitude ι, then projects this estimate onto a Fourier basis of the value function space with $F < n$ coefficients with respect to the ν-quadratic norm (projection that we write $\Pi_{F,\nu}$), and then applies the (exact) greedy operator on this projected estimate. In a nutshell, one call to the approximate greedy operator $\mathcal{G}_\epsilon(\nu, v)$ amounts to compute $\mathcal{G}\Pi_{F,\nu}(v + u(\iota))$.

[6] We implemented a crude line-search mechanism, that looks on the set $2^i\alpha$ where α is the minimal step estimated by CPI to ensure improvement.

Simulations We have run series of experiments, in which we callibrated the perturbations (noise, approximations) so that the algorithm are significantly perturbed but no too much (we do not want their behavior to become too erratic). After trial and error, we ended up considering the following setting. We used garnet problems $G(n_s, n_a, b, p)$ with the number of states $n_s \in \{100, 200\}$, the number of actions $n_a \in \{2, 5\}$, the branching factor $b \in \{1, \frac{n_s}{50}\}$ ($b = 1$ corresponds to deterministic problems), the number of features to approximate the value $p = \frac{n_s}{10}$, and the noise level $\iota = 0.05$ (5%).

For each of these $2^3 = 8$ parameter instances, we generated 30 (random) MDPs. For each such MDP, we ran DPI, CPI+, CPI(0.1) and NSDPI 30 times and estimated (for all iterations) the average and the standard deviation (on these 30 runs for a fixed MDP) of the performance (the error $\mu(v_{\pi_*} - v_{\pi_k})$). Figures 2 to 5 display statistics of the average performance of the algorithms and of their standard deviation: statistics are here used because the average and the standard performance of an algorithm on an MDP are random variables since the MDP is itself random. For ease of comparison, all curves are displayed with the same x and y range. Figure 2 shows the statistics overall for the $2^3 = 8$ parameter instances. Figure 3, 4 and 5 display statistics that are respectively conditional on the values of n_s, n_a and b, which also gives some insight on the influence of these parameters.

From these experiments and statistics, we made the following general observations: 1) In all experiments, CPI+ converged in very few iterations (most of the time less than 10, and always less than 20). 2) DPI is much more variable than the other algorithms and tends to provide the worst average performance. 3) If CPI+ and NSDPI have a similar average performance, the standard deviation of NSDPI is consistantly smaller than that of CPI and is thus more robust. 4) CPI(0.1) tend to provide the best average results, though its standard deviation is bigger than that of NSDPI. 5) All these relative behaviors tend to be amplified in the more difficult instances, that is when the state and actions spaces are big ($n_s = 200$, $n_a = 5$) and the dynamics deterministic ($b = 1$).

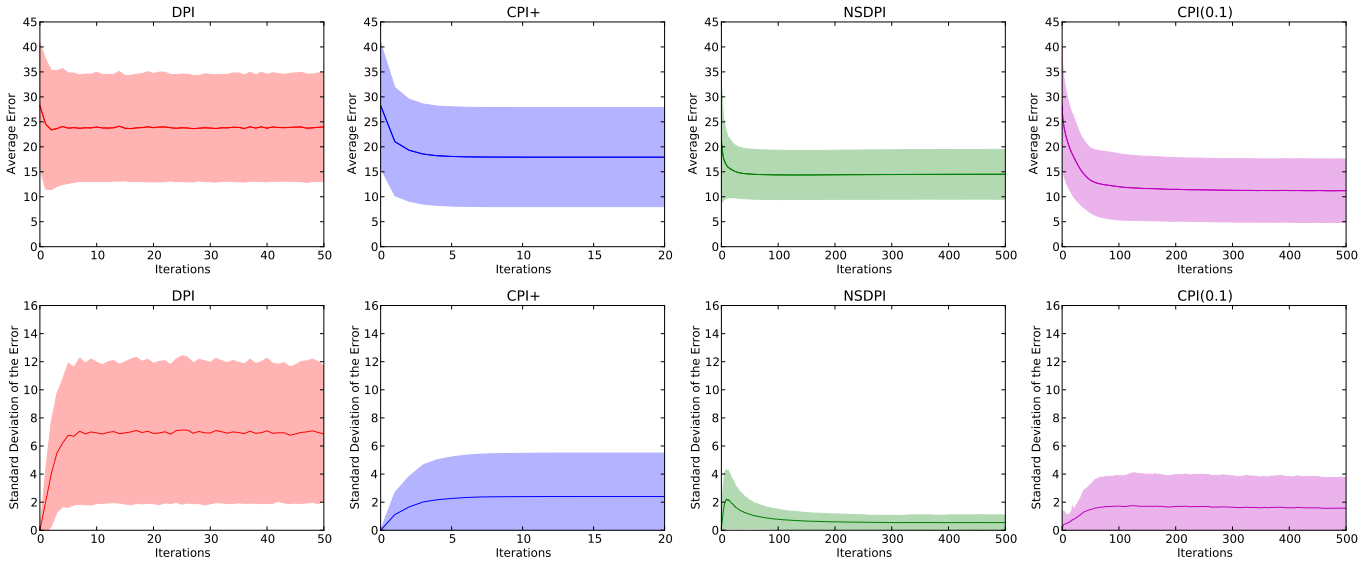
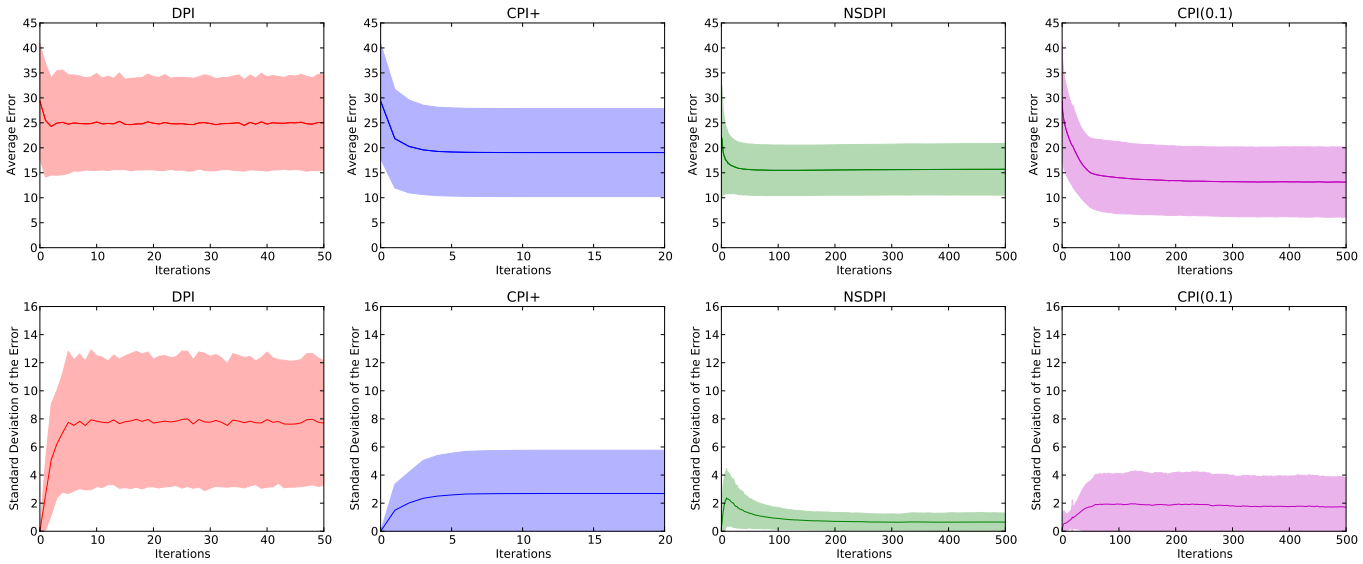


Figure 2: Statistics for all instances

$n_s = 100$



$n_s = 200$

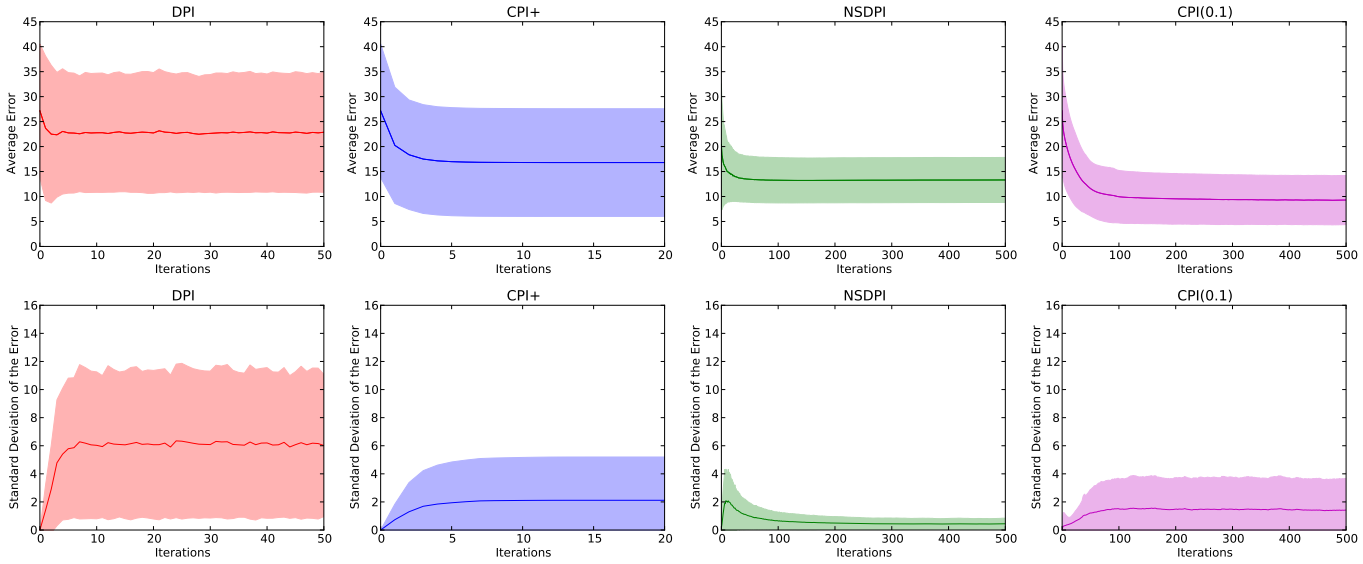
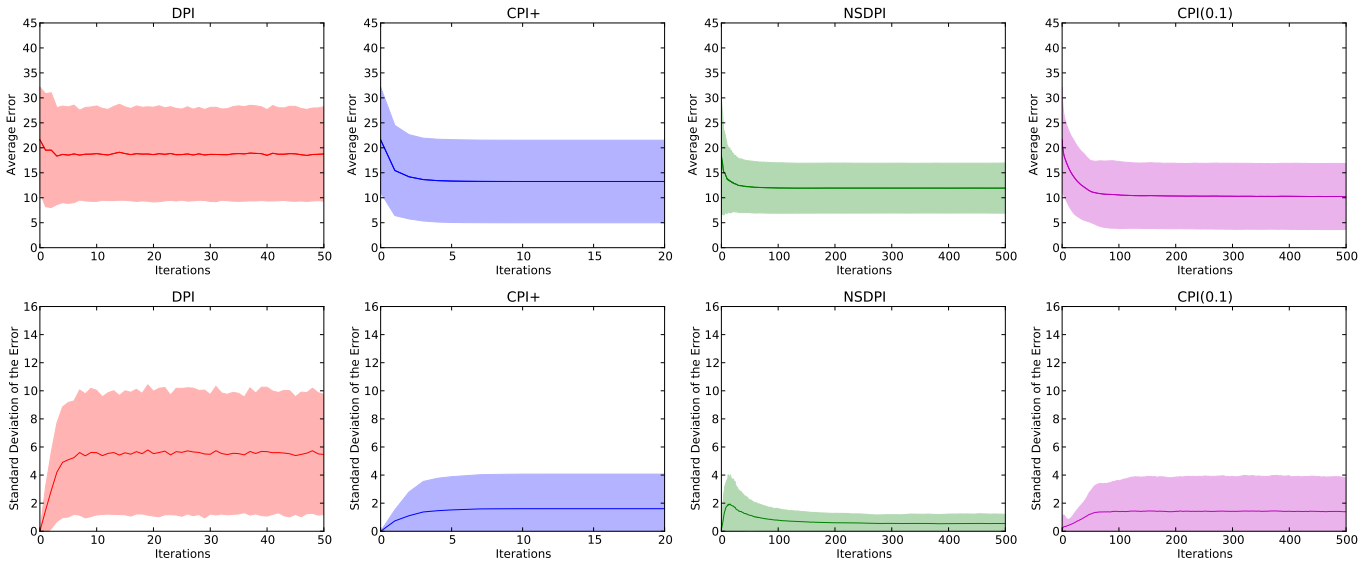


Figure 3: Statistics conditioned on the number of states. Top: $n_s = 100$. Bottom: $n_s = 200$.

$n_a = 2$



$n_a = 5$

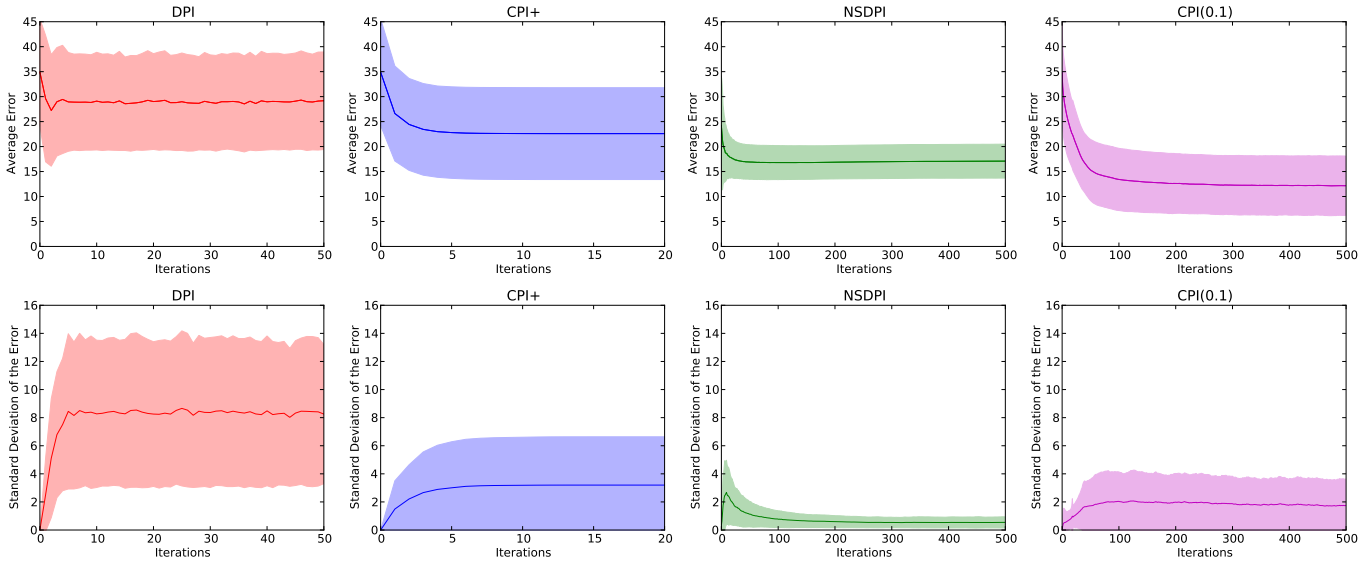
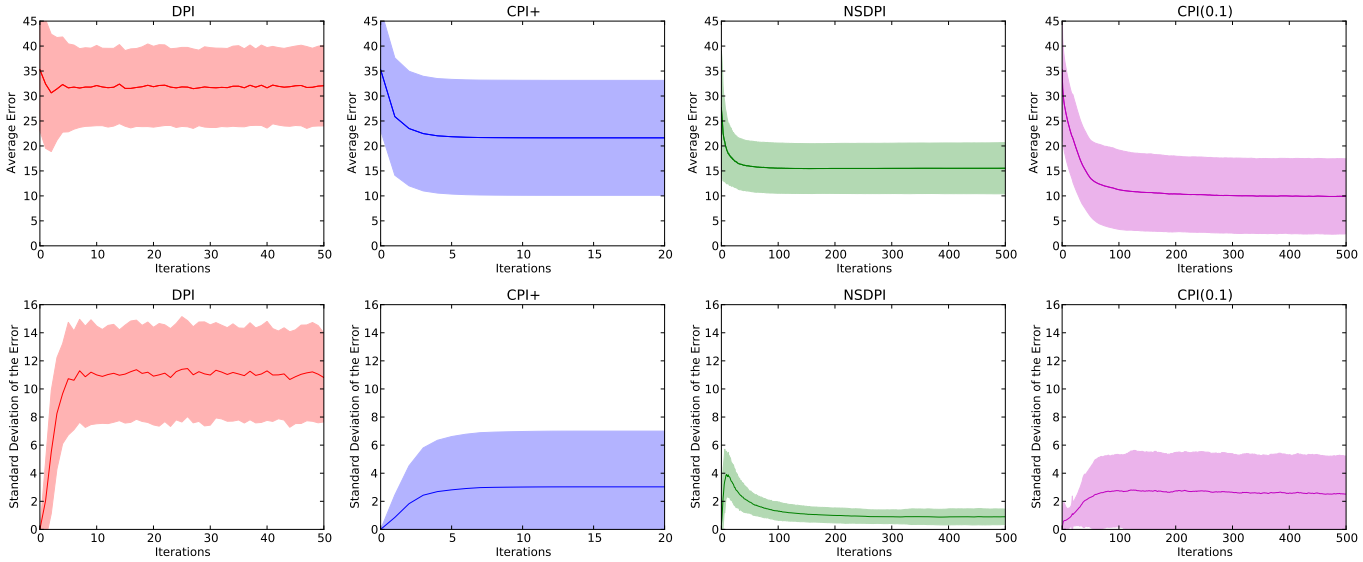


Figure 4: Statistics conditioned on the number of actions. Top: $n_a = 2$. Bottom: $n_a = 5$.

$b = 1$ (deterministic)



$b = \frac{n_a}{50}$

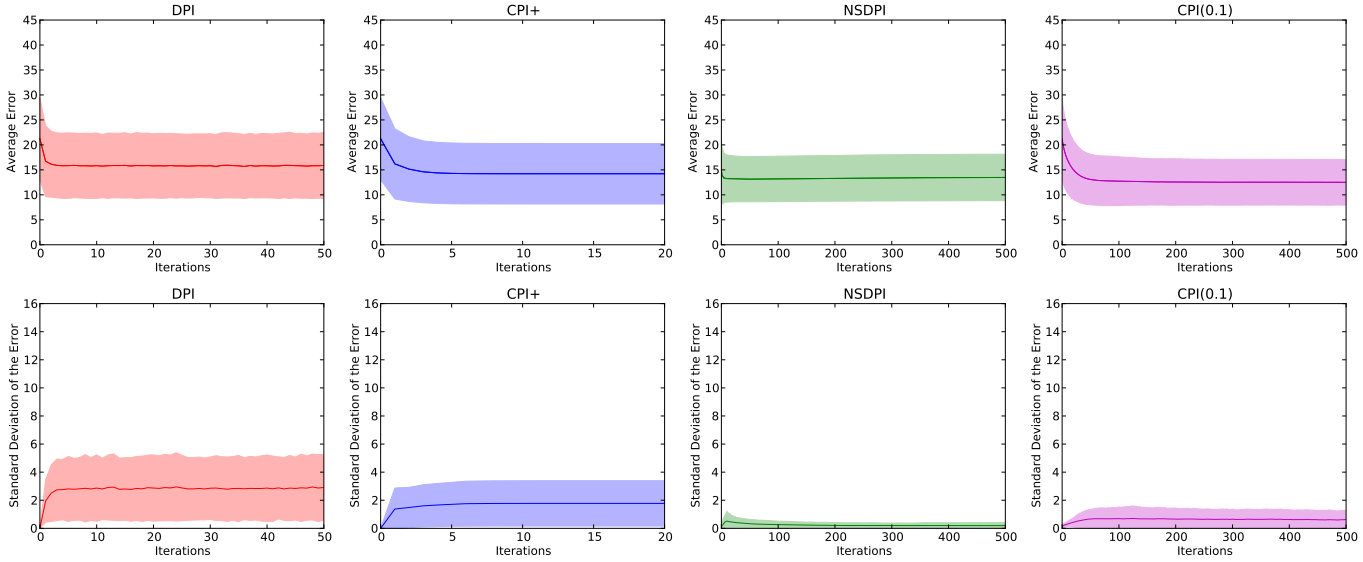


Figure 5: Statistics conditioned on the branching factor. Top: $b = 1$ (deterministic). Bottom: $b = \frac{n_a}{50}$.