Local Policy Search in a Convex Space and Conservative Policy Iteration as Boosted Policy Search

Bruno Scherrer[1], Matthieu Geist[2]

[1] Inria, Villers-lès-Nancy, F-54600, France,
Université de Lorraine, LORIA, UMR 7503, Vandœuvre-lès-Nancy, F-54506, France
[2] Supélec – IMS-MaLIS Research Group & UMI 2958 (GeorgiaTech-CNRS), Metz, France

Abstract. Local Policy Search is a popular reinforcement learning approach for handling large state spaces. Formally, it searches locally in a parameterized policy space in order to maximize the associated value function averaged over some predefined distribution. The best one can hope in general from such an approach is to get a local optimum of this criterion. The first contribution of this article is the following surprising result: if the policy space is convex, *any* (approximate) *local optimum* enjoys a *global performance guarantee*. Unfortunately, the *convexity* assumption is strong: it is not satisfied by commonly used parameterizations and designing a parameterization that induces this property seems hard. A natural solution to alleviate this issue consists in deriving an algorithm that solves the local policy search problem using a boosting approach (constrained to the convex hull of the policy space). The resulting algorithm turns out to be a slight generalization of conservative policy iteration; thus, our second contribution is to highlight an original connection between local policy search and approximate dynamic programming.

1 Introduction

We consider the reinforcement learning problem [24] formalized through Markov Decision Processes (MDP) [21], in the situation where the state space is large and approximation is required. On the one hand, Approximate Dynamic Programming (ADP) is a standard approach for handling large state spaces. It consists in mimicking in an approximate form the standard algorithms that were designed to optimize globally the policy (maximizing the associated value function for each state). On the other hand, Local Policy Search (LPS) consists in parameterizing the policy (often called an "actor") and locally maximizing the associated expected value function. This can be done for example using a (natural) gradient ascent [3, 10]—possibly with a critic [25, 20], expectation-maximization (EM) [12], or even directly using some black-box optimization algorithm [9]. LPS methods work particularly well in practice: the just cited papers describe applications to standard benchmarks and applications such as robotics, that are competitive with the ADP approach. Surprisingly, gradient-based and EM approaches, that are usually prone to be stuck in local optima, do not seem to be penalized in applications to Reinforcement Learning. Even more surprisingly, it was shown [10] that a natural gradient ascent in the policy space can outperform ADP on the Tetris game.

Following the seminal works by [4], it has been shown that ADP algorithms enjoy global performance guarantees, bounding the loss of using the computed policy instead of using the optimal one as a function of the approximation errors involved along the iterations: see [18] for approximate policy iteration (API), [19] for approximate value iteration (AVI), or more generally [22] for approximate modified policy iteration. To the best of our knowledge, similar general guarantees do not exist in the literature for LPS algorithms. In general though, the best one can hope for LPS is to get a local optimum of the optimized fitness (that is, a local maximum of the averaged value function), and the important question of the loss with respect to the optimal policy remains open. As for instance mentioned as the main "future work" in [6], where the convergence of a family of natural actor-critic algorithms is proven, "*[i]t is important to characterize the quality of converged solutions.*" The motivation of this paper is to deepen the understanding on the LPS approach.

Our main contribution (Theorem 3, Section 3) is to show that if the policy space on which one performs LPS is a convex subset of the full space of stochastic policies—equivalently this means that if two policies are taken in the space, then their stochas-

s-

s-insthenTss

is convex (or equivalently stable by stochastic mixture) if it satisfies:

$$\forall \pi, \pi' \in \Pi, \quad \forall \alpha \in (0,1), \quad (1-\alpha)\pi + \alpha\pi' \in \Pi.$$

For a given policy π, we define $r_\pi \in \mathbb{R}^{\mathcal{S}}$ as

$$r_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) r(s,a) = \mathbb{E}_{a \sim \pi(\cdot|s)}[r(s,a)]$$

and $P_\pi \in (\Delta_{\mathcal{S}})^{\mathcal{S}}$ as

$$P_\pi(s'|s) = \sum_{a \in \mathcal{A}} \pi(a|s) P(s'|s,a) = \mathbb{E}_{a \sim \pi(\cdot|s)}[P(s'|s,a)].$$

The value function v_π quantifies the quality of a policy π for each state s by measuring the expected cumulative reward received for starting in this state and then following the policy:

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

The Bellman operator T_π of policy π associates to each function $v \in \mathbb{R}^{\mathcal{S}}$ the function defined as

$$[T_\pi v](s) = \mathbb{E}\left[r_\pi(s) + \gamma v(s') | s' \sim P_\pi(\cdot|s)\right],$$

or more compactly $T_\pi v = r_\pi + \gamma P_\pi v$. The value function v_π is known to be the unique fixed point of T_π.

It is also well-known that there exists a policy π_* that is optimal in the sense that it satisfies $v_{\pi_*}(s) \geq v_\pi(s)$ for all states s and policies π. The value function v_* is the unique fixed point of the following nonlinear Bellman equation:

$$v_* = T v_* \text{ with } Tv = \max_{\pi \in \mathcal{A}} T_\pi v$$

where the $\max$ is taken componentwise. Given any function $v \in \mathbb{R}^{\mathcal{S}}$, we say that a policy π' is greedy with respect to v if $T_{\pi'} v = Tv$, and we write $\mathcal{G}(\pi)$ for the set of policies that are greedy with respect to the value v_π of some policy π. The notions of optimal value function and greedy policies are fundamental to optimal control because of the following property: any policy π_* that is greedy with respect to the optimal value is an optimal policy and its value v_{π_*} is equal to v_*. Therefore, an equivalent characterization of the optimality of some policy π is that it is greedy with respect to its own value:

$$\pi \in \mathcal{G}(\pi). \tag{1}$$

For any distribution μ, we define the γ-weighted occupancy measure[4] induced by the policy π when the initial state is sampled from μ as $d_{\mu,\pi} = (1-\gamma)\mu(I - \gamma P_\pi)^{-1}$ (we recall μ to be a row vector by convention) with $(I - \gamma P_\pi)^{-1} = \sum_{t \geq 0} (\gamma P_\pi)^t$.

[4] When it exists, this measure tends to the stationary distribution of P_π when the discount factor tends to 1.

It can easily be seen that $\mu v = \frac{1}{1-} d \; ; \; r$. For any two distributions μ and ν, we write $\left\| - \right\|_\infty$ for the smallest constant C satisfying $\mu(s) \leq C\nu(s)$, for any $s \in S$ (this constant is actually the supremum norm of the componentwise ratio, thus the notation).

From an algorithmic point of view, Dynamic Programming methods compute the optimal value policy pair (v_*, π_*) in an iterative way. When the problem is large and cannot be solved exactly, Approximate Dynamic Programming (ADP) refers to noisy implementations of these exact methods, where the noise is due to approximations at each iteration. For instance, Approximate Value and Policy Iteration respectively correspond to the following schemes:

$$v_{k+1} = Tv_k + \epsilon_k \quad \text{and} \quad \begin{cases} v_k = v_{\;k} + \epsilon_k \\ \pi_{k+1} \in \mathcal{G}(v_k) \end{cases} .$$

In the Local Policy Search (LPS) context on which we focus in this paper, we write Π the space where we perform the search. For a predefined distribution ν of interest, the problem addressed by LPS can be cast as follows:

$$\text{find } \pi \in \Pi \text{ s.t. } \pi \text{ is a local maximum of } J(\pi) = \mathbb{E}_{s\sim}[v(s)].$$

Assume that we are able to (approximately) find such a locally optimal policy π. A natural question is: can we say something about the distance between the value of this policy v and that of the optimal policy $v_* = v_{\;*}$? Quite surprisingly, and in contrast with most optimization problems, we are going to provide a condition on the policy space Π that allows to give a nontrivial performance guarantee; this is the aim of the next section.

3 Main Result

In order to state our main result, we need to define a relaxation of the set of policies that are greedy with respect to some given policy.

Definition 1 (μ**-weighted** ϵ**-greedy policies).** *We write* $\mathcal{G}(\pi, \mu, \epsilon)$ *for the set of policies which are* ϵ*-greedy respectively to* π *(in* μ*-expectation), formally defined as*

$$\mathcal{G}(\pi, \mu, \epsilon) = \{\pi' \in \Pi \text{ such that } \forall \pi'' \in \Pi,\ \mu T_{\pi'} v + \epsilon \geq \mu T_{\pi''} v\}.$$

This is indeed a relaxation of $\mathcal{G}$, as it can be observed that for all policies π and π',

$$\begin{aligned} \pi' \in \mathcal{G}(\pi) \quad &\Leftrightarrow \quad \forall \mu \in \Delta_{\mathcal{S}},\ \pi' \in \mathcal{G}(\pi, \mu, 0) \\ &\Leftrightarrow \quad \exists \mu \in \Delta_{\mathcal{S}},\ \mu > 0,\ \pi' \in \mathcal{G}(\pi, \mu, 0). \end{aligned}$$

We are now ready to state our first important result.

Theorem 1. *Let* π *be some policy in* Π*. The following two properties are equivalent:*

$$\forall \pi' \in \Pi, \quad \lim_{\to 0} \frac{\nu v_{(1-)+\pi'} - \nu v}{\alpha} \leq \epsilon. \tag{2}$$

$$\pi \in \mathcal{G}(\pi, d \; ; \; , (1-\gamma)\epsilon). \tag{3}$$

Equation (3) says that the policy π is approximately greedy with respect to itself, and can be thus seen as a relaxed version of the optimality Equation (1); as we will show below, this will allow us to provide a global performance guarantee for the policy π. Equation (2) says that π is an approximate local optimum of $\pi \mapsto J\ (\pi)$ if π is allowed to move in the convex hull of the policy space $\mathit{\Pi}$: indeed, whatever the direction we look at in this space, the slope of the improvement—locally around π—is bounded by ϵ. Theorem 1 thus has the following corollary.

Corollary 1 *Assume that the space $\mathit{\Pi}$ is convex. Then any policy π that is an ϵ-local optimum of $\pi \mapsto J\ (\pi)$ (in the sense of Equation* (2)*) satisfies the relaxed Bellman Equation* (3)*.*

We now turn to the proof of Theorem 1. The following technical (but simple) lemma will be useful for the proof.

Lemma 1. *For any policies π and π', we have*

$$v^{\pi'} - v^{\pi} = (I - \gamma P^{\pi'})^{-1}(T^{\pi'} v^{\pi} - v^{\pi}).$$

Proof. The proof uses the fact that the linear Bellman Equation $v^{\pi} = r^{\pi} + \gamma P^{\pi} v^{\pi}$ implies $v^{\pi} = (I - \gamma P^{\pi})^{-1} r^{\pi}$. Then,

$$\begin{aligned} v^{\pi'} - v^{\pi} &= (I - \gamma P^{\pi'})^{-1} r^{\pi'} - v^{\pi} \\ &= (I - \gamma P^{\pi'})^{-1}(r^{\pi'} + \gamma P^{\pi'} v^{\pi} - v^{\pi}) \\ &= (I - \gamma P^{\pi'})^{-1}(T^{\pi'} v^{\pi} - v^{\pi}). \end{aligned}$$

□

Proof (Proof of Theorem 1). For any α and any $\pi' \in \mathit{\Pi}$, write $\pi^{\alpha} = (1-\alpha)\pi + \alpha\pi'$. Using Lemma 1, we have:

$$\nu(v^{\pi_\alpha} - v^{\pi}) = \nu(I - \gamma P^{\pi_\alpha})^{-1}(T^{\pi_\alpha} v^{\pi} - v^{\pi}).$$

By observing that $r^{\pi_\alpha} = (1-\alpha) r^{\pi} + \alpha r^{\pi'}$ and $P^{\pi_\alpha} = (1-\alpha) P^{\pi} + \alpha P^{\pi'}$, it can be seen that $T^{\pi_\alpha} v^{\pi} = (1-\alpha) T^{\pi} v^{\pi} + \alpha T^{\pi'} v^{\pi}$. Thus, using the fact that $v^{\pi} = T^{\pi} v^{\pi}$, we get:

$$\begin{aligned} T^{\pi_\alpha} v^{\pi} - v^{\pi} &= (1-\alpha) T^{\pi} v^{\pi} + \alpha T^{\pi'} v^{\pi} - v^{\pi} \\ &= \alpha (T^{\pi'} v^{\pi} - v^{\pi}). \end{aligned}$$

In parallel, we have

$$\begin{aligned} (I - \gamma P^{\pi_\alpha})^{-1} &= (I - \gamma P^{\pi} + \alpha\gamma (P^{\pi} - P^{\pi'}))^{-1} \\ &= (I - \gamma P^{\pi})^{-1} (I + \alpha M), \end{aligned}$$

where M is bounded (the exact form of the matrix M does not matter). Put together, we obtain

$$\nu(v^{\pi_\alpha} - v^{\pi}) = \alpha \nu (I - \gamma P^{\pi})^{-1}(T^{\pi'} v^{\pi} - v^{\pi}) + O(\alpha^2).$$

Taking the limit, we obtain

$$\lim_{\to 0} \frac{\nu(v_{\ \alpha} - v\ )}{\alpha} = \nu(I - \gamma P\ )^{-1}(T\ ' v\ - v\ )$$
$$= \frac{1}{1-\gamma} d\ ;\ (T\ ' v\ - v\ ),$$

and the result follows.

A second important step in our analysis consists in showing that a relaxed optimality characterization as the one of Equation (3) implies a global performance guarantee. To state this result, we first need to define the "ν-greedy-complexity" of our policy space, which measures how good Π was designed so as to approximate the greedy operator, for a starting distribution ν.

Definition 2 (ν-greedy-complexity). *We define $\mathcal{E}\ (\Pi)$ the ν-greedy-complexity of the policy space Π as*

$$\mathcal{E}\ (\Pi) = \max_{\in} \min_{'\in} \left(d\ ;\ (Tv\ - T\ ' v\ )\right).$$

Since $Tv\ - T\ v\ = Tv\ - v\ \geq 0$, we have $\mathcal{E}\ (\Pi) \geq 0$ for any policy space Π. In the limit case where Π contains all (deterministic) policies, we have $\mathcal{E}\ (\Pi) = 0$.

Given this definition, we are ready to state our second important result.

Theorem 2. *If $\pi \in \mathcal{G}\ (\pi, d\ ;\ , \epsilon)$, then for any policy π' and for any distribution μ over $\mathcal{S}$, we have*

$$\mu v\ ' \leq \mu v\ + \frac{1}{(1-\gamma)^2} \left\| \frac{d\ ;\ '}{\nu} \right\|_\infty (\mathcal{E}\ (\Pi) + \epsilon).$$

Notice that this theorem is actually a slight[5] generalization of Theorem 6.2 of [11]. We provide the proof for the sake of completeness.

Proof. Using again Lemma 1 and the fact that $Tv\ \geq T\ ' v\ $, we have

$$\mu(v\ ' - v\ ) = \mu(I - \gamma P\ ')^{-1}(T\ ' v\ - v\ )$$
$$= \frac{1}{1-\gamma} d\ ;\ '(T\ ' v\ - v\ ) \leq \frac{1}{1-\gamma} d\ ;\ '(Tv\ - v\ ).$$

Since $Tv\ - v\ \geq 0$ and $d\ ;\ \geq (1-\gamma)\nu$, we get

$$\mu(v\ ' - v\ ) \leq \frac{1}{1-\gamma} \left\| \frac{d\ ;\ '}{\nu} \right\|_\infty \nu(Tv\ - v\ )$$
$$\leq \frac{1}{(1-\gamma)^2} \left\| \frac{d\ ;\ '}{\nu} \right\|_\infty d\ ;\ (Tv\ - v\ ).$$

[5] Theorem 2 holds for any policy π', not only for the optimal one, and the error term is split up (which is necessary to provide a more general result).

Using $d\ ;\ (Tv\ -v\ ) = (d\ ;\ Tv\ -d\ ;\ v\ )$, we get

$$\mu(v\ ' - v\ ) \leq \frac{1}{(1-\gamma)^2} \left\| \frac{d\ ;\ '}{\nu} \right\|_\infty \times$$
$$\left(d\ ;\ Tv\ - \max_{'\in} d\ ;\ T\ 'v\ + \max_{'\in} d\ ;\ T\ 'v\ - d\ ;\ v\ \right)$$
$$\leq \frac{1}{(1-\gamma)^2} \left\| \frac{d\ ;\ '}{\nu} \right\|_\infty (\mathcal{E}\ (\Pi) + \epsilon). \qquad \square$$

The first main result of the paper is a straightforward combination of Corollary 1 and Theorem 2.

Theorem 3. *Assume that the space Π is convex. Then any policy π that is an ϵ-local optimum of $\pi \mapsto J\ (\pi)$ (in the sense of Equation (2)) enjoys the following global performance guarantee:*

$$\mathbb{E}_{\mathsf{S}\sim}\ [v_*(s) - v\ (s)] \leq \frac{1}{1-\gamma} \left\| \frac{d\ ;\ *}{\nu} \right\|_\infty \left(\frac{\mathcal{E}\ (\Pi)}{1-\gamma} + \epsilon \right).$$

4 About the Convex Policy Space Assumption

The remarkable result of the previous section—a connection between *local* optimality and *global* guarantee—relies on the assumption that the policy space Π is convex. Though this assumption may look mild at first sight, we are going to argue that it is in fact strong. We will then propose a natural algorithmic approach for performing Local Policy Search on the convex hull of some (not necessarily convex) policy space Π.

4.1 A Strong Assumption

A common approach (for continuous actions mainly) is to parameterize a mapping from state to actions and to put it as the mean of a Gaussian distribution, that is

$$\pi\ (a|s) \propto \exp\left(-\frac{1}{2} \left\| a - u\ (s) \right\|^2_{\ -1} \right),$$

with here u the parameterized state to action mapping and Σ a predefined covariance matrix. Obviously, the space of such policies is not convex, since a mixture of Gaussian distributions is in general not a Gaussian distribution. Another common approach (for discrete actions) is to adopt a parameterized Gibbs distribution, that is

$$\pi\ (a|s) \propto \exp\left(\theta^\top \psi(s,a) \right),$$

where $\theta^\top \psi(s,a)$ can be seen as a parameterized state-action or score function. Here again, the resulting policy space is not convex in general.

In fact, we consider that it is an open problem to design a non-trivial parameterization that defines a convex policy space (by non-trivial, we mean a space that is neither simply a convex combination of a *small* number of policies nor the *full* convex hull of $\mathcal{A}^\mathcal{S}$). Even in a one-state situation, the answer does not seem obvious: this requires to find distributions that are stable by mixture and we did not manage to find any satisfying solution. An alternative approach, that we develop next, is to consider for Π the convex hull of a set of parameterized policies.

4.2 Boosting

Let $\mathcal{P}$ be a space of policies and $\Pi = \mathrm{co}(\mathcal{P})$ denote its convex hull. We propose to use boosting for finding a local maximum of $J\ (\pi)$ on Π. More precisely, we propose to apply the AnyBoost.L1 algorithm [17]: it sees boosting as a gradient ascent in function space and constrains the search in the convex hull of the base policy space. Let $\nabla J\ (\pi)$ be the functional gradient (according to π) of the LPS objective function. Applied to our problem, AnyBoost.L1 works as follows. At iteration k, we have a policy π_{k-1}, and perform the following steps:

1. compute $h_k \in \mathrm{argmax}_{h\in\mathcal{P}}\langle \nabla J\ (\pi_{k-1}), h\rangle$,
2. update the policy: $\pi_k = (1-\alpha_k)\pi_{k-1} + \alpha_k h_k$, with $\alpha_k \in (0,1)$ the learning rate.

The basic idea is to perform a functional gradient ascent on $J\ (\pi)$. However, the gradient $\nabla J\ (\pi_{k-1})$ does not generally belong to $\mathcal{P}$, so we search for a policy h with greatest inner product with $\nabla J\ (\pi_{k-1})$. This corresponds to the first step. The second step updates the policy as a mixture of the old one and of the computed h_k, the mixture weight α_k being the learning rate of the gradient ascent. In order to obtain a more practical algorithm, one has to rephrase the optimization problem of the first step.

Proposition 1 *We have that*

$$\operatorname*{argmax}_{h\in\mathcal{P}} \langle \nabla J(\pi), h\rangle = \operatorname*{argmin}_{h\in\mathcal{P}} d\ ;\ (Tv\ - T_h v\ ).$$

In particular, assume that $\mathcal{P}$ is a space of deterministic policies and define $q\ = T_a v$ the state-action value function of a policy π (writing with a slight abuse of notation T_a the Bellman operator for the policy associating action a to any state), then

$$\operatorname*{argmax}_{h\in\mathcal{P}} \langle \nabla J(\pi), h\rangle = \operatorname*{argmin}_{h\in\mathcal{P}} \sum_{s\in\mathcal{S}} d\ ;\ (s)\left(\max_{a\in\mathcal{A}} q\ (s,a) - q\ (s,h(s))\right).$$

This process can be seen as an approximate version of the greedy step of the Policy Iteration algorithm and may be implemented through a weighted classification problem, or through an ℓ_p-regression of the $q\ $ function.

Proof (Proof of Proposition 1). The functional gradient of $J\ $ is

$$\nabla J\ (\pi) = \frac{1}{1-\gamma}\sum_{s\in\mathcal{S}} d\ ;\ (s) \sum_{a\in\mathcal{A}} \nabla\pi(a|s) q\ (s,a).$$

This is a rather direct extension of the classic policy gradient theorem [25]. Then, we need to compute its inner product with a function h of $\mathcal{P}$:

$$\begin{aligned}
\langle \nabla J(\pi), h\rangle &= \frac{1}{1-\gamma}\langle \sum_s d\ ;\ (s) \sum_a \nabla\pi(a|s) q\ (s,a), h\rangle \\
&= \frac{1}{1-\gamma}\sum_s d\ ;\ (s) \sum_a \langle \nabla\pi(a|s), h\rangle q\ (s,a) \\
&= \frac{1}{1-\gamma}\sum_s d\ ;\ (s) \sum_a h(a|s) q\ (s,a) \\
&= \frac{1}{1-\gamma} d\ ;\ (T_h v\ ).
\end{aligned}$$

Eventually, this allows concluding:

$$\begin{aligned}\operatorname*{argmax}_{h\in\mathcal{H}}\langle\nabla J, h\rangle &= \operatorname*{argmax}_{h\in\mathcal{H}} d\ ;\ (T_h v\ \ ) \\ &= \operatorname*{argmin}_{h\in\mathcal{H}} d\ ;\ (Tv\ \ - T_h v\ \ ).\end{aligned}$$

□

4.3 Connection to CPI

Thus, the boosting approach to LPS consists in computing a mixture of policies, each new component of the mixture being the solution of an approximation of the greedy policy respectively to the preceding estimated mixture. It turns out that Conservative Policy Iteration (CPI) [11] is a specific case of this general algorithm, the only difference being that CPI chooses specific values for the learning rate (such as guaranteeing improvements).

If the algorithm resulting from this boosting approach is not really new, it provides some clarifications about LPS, API and CPI. First, this shows that CPI can be derived as an LPS approach, whereas it was originally derived from an API viewpoint, with the desire to fix the potential policy degradation problem of API [11]. This draws a connection between API and LPS that has not yet been documented in the literature, and highlights the fact that CPI is at the frontier of these two approaches. Second, it provides some leads of improvement for CPI (which has strong guarantees but is in general slow). One could also choose the learning rates according to the boosting optimization theory, or use related heuristics or even some line search. Last but not least, AnyBoost.L1 is perhaps the more natural way to search for a local maximum of J on a convex policy space. Looking for alternative algorithms performing LPS in convex policy spaces is an interesting research direction.

5 Discussion

In this section, we discuss the relations of our analyses with previous works, we compare this guarantee with the standard ones of approximate dynamic programming (focusing particularly on the API algorithm) and we discuss some practical and theoretical consequences of our analysis.

5.1 Closely Related Analysis

A performance guarantee very similar to the one we provide in Theorem 2 was first derived for CPI by [11]. This result of the literature was certainly considered specific to the CPI algorithm, that has unfortunately not been used widely in practice probably because of its somewhat complex implementation. In contrast, we show in this paper that such a performance guarantee is valid for any method that finds a policy that satisfies a relaxed Bellman identity like that given Equation (3), among which CPI naturally arises, as shown in section 4.

Though the main result of our paper is Theorem 3, and since Theorem 2 appears in a very close form in [11], our main technical contribution is Theorem 1 that highlights

a deep connection between local optimality and a relaxed Bellman optimality characterization. A result, that is similar in flavor, is derived by [10] for the Natural Policy Gradient algorithm: Theorem 3 there shows that natural gradient updates are moving the policy towards the solution of a (DP) update. The author writes: "*The natural gradient could be efficient far from the maximum, in that it is pushing the policy toward choosing greedy optimal actions*". Though there is an obvious connection with our work, the result there is limited since—similarly to the work we have just mentioned on CPI—*(i)* it seems to be specific to the natural gradient approach (though our result is general), and *(ii)* it is not exploited so as to connect with a global performance guarantee.

5.2 Relations to Bounds of Approximate Dynamic Programming

The performance guarantee of any approximate dynamic programming algorithm implies *(i)* a (quadratic) dependency on the average horizon $\frac{1}{1-}$, *(ii)* a concentration coefficient (which quantifies the divergence between the worst discounted average future state distribution when starting from the measure of interest, and the distribution used to control the estimation errors), and *(iii)* an error term linked to the estimation error encountered at each iteration (which can be due to the approximation of value functions and/or policies). Depending on what quantity is estimated, a comparison of these estimation errors may be hard. To ease the comparison, the following discussion focuses on the API algorithm. Note however that several aspects of our comparison holds for other ADP algorithms.

API generates a sequence of policies: at each iteration, a new policy is one that is approximately greedy with respect to the value of the previous policy. This can be achieved through an ℓ_p-regression of the state-action value function [5, 18, 13] or through a weighted classification problem [14, 7, 16]. Whatever the approach, the sequence of policies belongs (implicitely for ℓ_p-regression or explicitely for classification) to some space $\mathcal{P}$ that is typically a set of *deterministic policies*. For an initial policy π_0 and a given distribution ν, the API algorithm iterates as follows:

$$\begin{gathered}\text{pick } \pi_{k+1} \in \mathcal{P} \\ \text{such as (approximately) minimizing } \nu(Tv_{\ k} - T_{\ k+1} v_{\ k}).\end{gathered}$$

This is similar to CPI/boosted LPS, up to the fact that *(i)* it uses ν instead of $d_{\ ;}$ to approximate the greedy policy and *(ii)* it is optimistic (in the sense that $\alpha_k = 1$). To provide the API bound, we need an alternative concentration coefficient as well as some new error characterizing the quality of the space $\mathcal{P}$. Let $C_{\ ;}$ be the concentration coefficient defined as

$$C_{\ ;} = (1-\gamma)^2 \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} \gamma^{i+j} \sup_{\in \mathcal{A}} \left\| \frac{\mu (P_{\ *})^i (P_{\ })^j}{\nu} \right\|_\infty .$$

Consider the measure of the complexity of the policy space $\mathcal{P}$, similar to $\mathcal{E}$:

$$\mathcal{E}'(\mathcal{P}) = \max_{\in \mathcal{P}} \min_{'\in \mathcal{P}} (\nu(Tv_{\ } - T_{\ '} v_{\ })).$$

Let also e be an estimation error term that tends to zero as the number of samples tends to infinity (at a rate depending on the chosen approximator). The performance guarantee of API [18, 1, 15, 16, 8] can be expressed as follows:

$$\limsup_{k\to\infty} \mu(v^* - v_{\quad k}) \leq \frac{C_{\ ;}}{(1-\gamma)^2}(\mathcal{E}'(\mathcal{P}) + e).$$

This bound is to be compared with the result of Theorem 3, regarding the three terms involved: the average horizon, the concentration coefficient and the greedy error term. Each term is discussed now, a brief summary being provided in Table 1. As said in Section 5.1, the LPS bound is really similar to the CPI one, and the bounds of CPI and a specific instance of API have been compared by [8]. Our discussion can be seen as complementary: we consider API more generally, we provide some new elements of comparison, and we illustrate the methods empirically.

Table 1. Comparison of the performance guarantees for LPS and API.

	bounded term	horizon term	concentration term	error term
LPS	$\mu(v_* - v_\pi)$	$\frac{1}{(1-\gamma)^2}$	$\left\|\frac{d_{\mu,\pi_*}}{\nu}\right\|_\infty$	$\mathcal{E}_\nu(\Pi) + \epsilon(1-\gamma)$
API	$\limsup_{k\to\infty} \mu(v^* - v_{\pi_k})$	$\frac{1}{(1-\gamma)^2}$	$C_{\mu,\nu}$	$\mathcal{E}'_\nu(\mathcal{P}) + e$

Horizon term. Both bounds have a quadratic dependency on the average horizon $\frac{1}{1-}$. For approximate dynamic programming, this bound can be shown to be tight [23], the only known solution to improve this being to introduce non-stationary policies [23]. The tightness of this bound for policy search is an open question. However, we suggest later in Section 5.3 a possible way to improve on this.

Concentration coefficients. Both bounds involve a concentration coefficient. They can be compared as follows.

Theorem 4. *We always have that:* $\left\|\frac{d_{\mu,\pi_*}}{}\right\|_\infty \leq \frac{1}{1-} C_{\ ;}$ *. Also, if there always exists a* ν *such that* $\left\|\frac{d_{\mu,\pi_*}}{}\right\|_\infty < \infty$ *(by choosing* $\nu = d_{\ ; \ *}$*), there might not exist a* ν *such that* $C_{\ ;} < \infty$*.*

Proof. Consider the inequality of the first part. By using the definition of $d_{\ ; \ *}$ and eventually the fact that $d_{\ ; \ *} \geq (1-\gamma)\nu$, we have

$$\begin{aligned}
C_{\ ;} &= (1-\gamma)^2 \sum_{i=0}^{\infty}\sum_{j=0}^{\infty} \gamma^{i+j} \sup_{\in(\ \ \mathcal{A})} \left\|\frac{\mu(P_{\ *})^i (P\ )^j}{\nu}\right\|_\infty \\
&\geq (1-\gamma)^2 \left\|\sum_{i;j=0}^{\infty} \gamma^{i+j} \frac{\mu(P_{\ *})^{i+j}}{\nu}\right\|_\infty = (1-\gamma)\left\|\sum_{i=0}^{\infty} \gamma^i \frac{d_{\ ; \ *}(P_{\ *})^i}{\nu}\right\|_\infty \\
&\geq (1-\gamma)^2 \left\|\sum_{i=0}^{\infty} \gamma^i \frac{\mu(P_{\ *})^i}{\nu}\right\|_\infty = (1-\gamma)\left\|\frac{d_{\ ; \ *}}{\nu}\right\|_\infty .
\end{aligned}$$

Let us concentrate on the second part. Consider an MDP with N states and N actions, with $\mu = \delta_1$ being a dirac on the first state, and such that from here action $a \in [1; N]$ leads in state a deterministically. Write $c = \sup_{\pi \in \mathcal{A}} \|\frac{P_\pi}{\cdot}\|_\infty$ the first term defining $C_{\cdot;\cdot}$. For any π, we have $\mu P^{\cdot} \leq c\nu$. Thus, for any action a we have $\delta_a \leq c\nu \Rightarrow 1 \leq c\nu(a)$. Consequently, $1 = \sum_{i=1}^{N} \nu(i) \geq \frac{1}{c} \sum_{i=1}^{N} 1 \Leftrightarrow c \geq N$. This being true for arbitrary $N \in \mathbb{N}$, we get $c = \infty$ and thus $C_{\cdot;\cdot} = \infty$. □

The second part of this result tells that we may have $\left\| \frac{d_{\mu,\pi_*}}{\cdot} \right\|_\infty \ll C_{\cdot;\cdot}$, which is clearly in favor of LPS (and CPI, which involves the same concentration as LPS).

Error terms. Both bounds involve an error term. The terms ϵ (LPS) and e (API) can be made arbitrarily small by increasing the computational effort (the time devoted to run the algorithm and the amount of samples used), though nothing more can be said in general without studying a specific algorithmic instance (*e.g.*, type of local search for LPS or type of regressor/classifier for API). The terms defining the "greedy complexity" of policy spaces can be partially compared. Because they use different distributions that can be compared ($d_{\cdot;\cdot} \geq (1-\gamma)\nu$), we have for all policy spaces Π [8],

$$\mathcal{E}'(\Pi) \leq \frac{\mathcal{E}(\Pi)}{1-\gamma}.$$

However, this result does not take into account the fact that LPS (or CPI for the discussion of [8]) works with *stochastic policies* while API works with *deterministic policies*. This make these terms not comparable in general.

Experiments. To get a more precise picture of the relative practical performance of API and LPS, we ran both algorithms on many randomly generated MDPs. In order to assess their quality, we consider finite problems where the exact value function can be computed. More precisely, we consider Garnet problems first introduced by [2], 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 value 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.

The algorithms API and LPS need to repeatedly compute $\mathcal{G}^{\cdot}$. In other words, they must be able to make calls to an approximate greedy operator applied to the value $v^{\cdot}$ of some policy π for some distribution ν or $d_{\cdot;\cdot}$. To implement this operator, we compute a noisy estimate of the value $v^{\cdot}$ with a uniform white noise $u(\iota)$ of amplitude ι, then project this estimate onto $\mathcal{H}$, the space spanned by the p chosen features, with respect to the μ-quadratic norm (projection that we write $\Pi_{\mathcal{H};\cdot}$), and then applies the (exact) greedy operator on this projected estimate. In a nutshell, one call to the approximate

greedy operator $\mathcal{G}\ (\pi, \mu, \epsilon)$ amounts to compute $\mathcal{G}(\Pi_{\mathcal{H};}\ (v\ + u(\iota)))$, with $\mu = \nu$ (API) or $\mu = d\ _{;}$ (LPS).

In our experiments, we consider Garnet problems with $n_s \in \{50, 100, 200\}$ states, with $n_a \in \{2, 5\}$ actions, and branching factors in $b \in \{1, 2, 10\}$. For each of the 2×3^2 resulting possible combinations, we generated 30 i.i.d. random MDPs $(M_i)_{1 \leq i \leq 30}$. For each such MDP M_i, we make 30 i.i.d. runs of *(i)* API and *(ii)* LPS with a gradient step-size of 0.1. For each run and algorithm, we compute the distance between the value of the output policy and that of the optimal policy $(\Delta_j)_{1 \leq j \leq 30}$. Figure 1 displays learning curves with statistics on these random variables. On this large set of problems, LPS

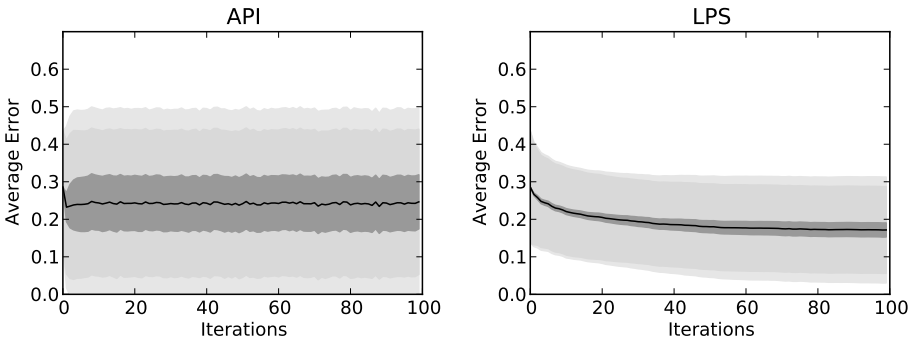

Fig. 1. Learning curves for API and LPS. MDPs are i.i.d. with the distribution of M_1. Conditioned on an MDP M_i, the error measures are i.i.d. with the distribution of Δ_1. The central line is an estimate of the overall average error $E[\Delta_1]$. The three grey regions (from dark to light) are estimates of the variability (across MDPs) of the average error $Std[E[\Delta_1|M_1]]$, the average (across MDPs) of the standard deviation of the error $E[Std[\Delta_1|M_1]]$, and the variability (across MDPs) of the standard deviation of the error $Std[Std[\Delta_1|M_1]]$.

significantly outperforms API, both on average and in terms of variability (across runs and problems). This confirms the importance of the better concentration coefficient of LPS, since it is in theory the main advantage of LPS over API.

5.3 Practical and Theoretical Consequences of our Analysis

Finally, this section provides a few important consequences of our analysis and of Theorem 3 in particular.

Rich policy and equivalence between local and global optimality. If the policy space is very rich, one can easily show that any local optimum is actually global (this result being a direct corollary of Theorem 3).

Theorem 5. *Let $\nu > 0$ be a distribution. Assume that the policy space is rich in the sense that $\mathcal{E}\ (\Pi) = 0$, and that π is an (exact) local optimum of $J\ (\epsilon = 0)$. Then, we have $v\ = v_*$.*

If this result is well-known in the case of tabular policies, it is to our knowledge new in such a general case (acknowledging that $\mathcal{E}\ (\Pi) = 0$ is a rather strong assumption).

Choice of the sampling distribution. Provided the result of Theorem 3, and as also mentioned about CPI by [11] since it satisfies a similar bound, if one wants to optimize the policy according to a distribution μ (that is, such that $\mu(v_* - v\ )$ is small), then one should optimize the fitness $J\ $ with the distribution $\nu \simeq d\ ;\ _*$ (so as to minimize the coefficient $\left\| \frac{d_{\mu,\pi_*}}{\ } \right\|_\infty$). Ideally, one should sample states based on trajectories following the optimal policy π_* starting from states drawn according to μ. This is in general not realistic since we do not know the optimal policy π_*, but practical solutions may be envisioned.

First, this means that one should sample states in the "interesting" part of the state space, that is where the optimal policy is believed to lead from the starting distribution μ. This is a natural piece of information that a domain expert should be able to provide. Also, though we leave the precise study of this idea for future research, a natural practical approach for setting the distribution ν would be to compute a sequence of policies $\pi_1, \pi_2, \ldots$ such that for all i, π_i is a local optimum of $\pi \mapsto J_{d_{\nu,\pi_{i-1}}}(\pi)$, that is of the criterion weighted by the region visited by the previous policy π_{i-1}. It may particularly be interesting to study whether the convergence of such an iterative process leads to interesting guarantees.

One may also notice that Theorem 3 may be straightforwardly written more generally for any policy. If π is an ϵ-local optimum of $J\ $ over $\mathit{\Pi}$, then for any stochastic policy π' we have

$$\mu v\ ' \leq \mu v\ \ + \frac{1}{1-\gamma} \left\| \frac{d\ ;\ '}{\nu} \right\|_\infty \left(\frac{\mathcal{E}\ (\mathit{\Pi})}{1-\gamma} + \epsilon \right).$$

Therefore, one can sample trajectories according to an acceptable (and known) controller π' so as to get state samples to optimize $J_{d_{\nu,\pi'}}$. More generally, if we know where a good policy π' leads the system to from some initial distribution μ, we can learn a policy π that is guaranteed to be approximately as good (and potentially better).

A better learning problem? With the result of Theorem 3, we have a squared dependency of the bound on the effective average horizon $\frac{1}{1-\ }$. For approximate dynamic programming, it is known that this dependency is tight [5, 23]. At the current time, this is an open question for policy search. However, we can improve the bound. We have shown that the ϵ-local optimality of a policy π implies that it satisfies a relaxed Bellman global optimality characterization, $\pi \in \mathcal{G}\ \ (\pi, d\ ;\ , \epsilon)$, which in turns implies Theorem 3. The following result, involving a slightly simpler relaxed Bellman equation, can be proved similarly to Theorem 2:

$$\text{If } \pi \in \mathcal{G}\ \ (\pi, \nu, \epsilon) \text{ then } \mu v\ ' \leq \mu v\ \ + \frac{1}{1-\gamma} \left\| \frac{d\ ;\ '}{\nu} \right\|_\infty (\mathcal{E}\ (\mathit{\Pi}) + \epsilon).$$

A policy satisfying this relaxed Bellman equation would have an improved dependency on the horizon ($\frac{1}{1-\ }$ instead of $\frac{1}{(1-\ )^2}$). At the current time, we do not know whether there exists an efficient algorithm for computing a policy satisfying $\pi \in \mathcal{G}\ \ (\pi, \nu, \epsilon)$. The above guarantee suggests that solving such a problem may improve over traditional policy search and approximate dynamic programming approaches.

6 Conclusion

In the past years, local policy search algorithms have been shown to be practical viable alternatives to the more traditional approximate dynamic programming field. The derivation of global performance guarantees for such approaches, probably considered as a desperate case, was to our knowledge never considered in the literature. In this article, we have shown a surprising result: *any Local Policy Search algorithm*, as long as it is able to *provide an approximate local optimum* of $J\ (\pi)$, *enjoys a global performance guarantee* similar to the ones of approximate dynamic programming algorithms. However, this relies on a strong convex policy space assumption, not satisfied by most standard local policy search algorithms. Weakening this hypothesis is an interesting research direction (yet difficult, as convexity is at the core of our analysis).

In order to handle this issue, we proposed to apply AnyBoost.L1 to local policy search. If it is a slight generalization of conservative policy iteration and is thus not a new algorithm, our work provides an original connexion between local policy search, boosting and approximate dynamic programming. Moreover, this suggests some open problems. First, AnyBoost.L1 (and thus CPI) is a rather natural approach to handle convex policy spaces. An interesting alternative would be to study the question of the parameterization of a convex space. If we were able to come up with a non-trivial parameterization, we could use many of the LPS algorithms of the literature (for instance actor-critic algorithms). Our analysis also suggests that it may be better to design algorithms that looks for a policy π satisfying $\pi \in \mathcal{G}\ (\pi, \nu, \epsilon)$ instead of searching for a local maximum of J , as it leads to a better bound (linear dependency on the average horizon). Working in that direction constitutes interesting future research. Last but not least, our experiments on Garnet problems showed that LPS outperforms API. Deepening the comparison of these approaches in larger problems constitutes natural future work.

References

1. Antos, A., Szepesvari, C., Munos, R.: Learning near-optimal policies with Bellman-residual minimization based fitted policy iteration and a single sample path. Machine Learning Journal 71, 89–129 (2008)
2. Archibald, T., McKinnon, K., Thomas, L.: On the Generation of Markov Decision Processes. Journal of the Operational Research Society 46, 354–361 (1995)
3. Baxter, J., Bartlett, P.L.: Infinite-horizon gradient-based policy search. Journal of Artificial Intelligence Research (JAIR) 15, 319–350 (2001)
4. Bertsekas, D., Tsitsiklis, J.: Neuro-Dynamic Programming. Athena Scientific (1996)
5. Bertsekas, D.P.: Dynamic Programming and Optimal Control. Athena Scientific (1995)
6. Bhatnagar, S., Sutton, R.S., Ghavamzadeh, M., Lee, M.: Incremental natural actor-critic algorithms. In: Advances in Neural Information Processing Systems (NIPS) (2007)
7. Fern, A., Yoon, S., Givan, R.: Approximate Policy Iteration with a Policy Language Bias: Solving Relational Markov Decision Processes. Journal of Artificial Intelligence Research (JAIR) 25, 75–118 (2006)
8. Ghavamzadeh, M., Lazaric, A.: Conservative and Greedy Approaches to Classification-based Policy Iteration. In: Conference on Artificial Intelligence (AAAI) (2012)

9. Heidrich-Meisner, V., Igel, C.: Evolution strategies for direct policy search. In: International Conference on Parallel Problem Solving from Nature (PPSN X). pp. 428–437 (2008)
10. Kakade, S.: A Natural Policy Gradient. In: Advances in Neural Information Processing Systems (NIPS) (2001)
11. Kakade, S., Langford, J.: Approximately optimal approximate reinforcement learning. In: International Conference on Machine Learning (ICML) (2002)
12. Kober, J., Peters, J.: Policy Search for Motor Primitives in Robotics. Machine Learning pp. 171–203 (2011)
13. Lagoudakis, M., Parr, R.: Least-squares policy iteration. Journal of Machine Learning Research (JMLR) 4, 1107–1149 (2003)
14. Lagoudakis, M., Parr, R.: Reinforcement learning as classification: Leveraging modern classifiers. In: International Conference on Machine Learning (ICML) (2003)
15. Lazaric, A., Ghavamzadeh, M., Munos, R.: Finite-sample analysis of least-squares policy iteration. Journal of Machine learning Research 13, 3041–3074 (2011)
16. Lazaric, A., Ghavamzadeh, M., Munos, R.: Analysis of a classification-based policy iteration algorithm. In: International Conference on Machine Learning (ICML) (2010)
17. Mason, L., Baxter, J., Bartlett, P., Frean, M.: Boosting algorithms as gradient descent in function space. Tech. rep., Australian National University (1999)
18. Munos, R.: Error bounds for approximate policy iteration. In: International Conference on Machine Learning (ICML) (2003)
19. Munos, R.: Performance bounds in Lp norm for approximate value iteration. SIAM Journal on Control and Optimization (2007)
20. Peters, J., Schaal, S.: Natural Actor-Critic. Neurocomputing 71, 1180–1190 (2008)
21. Puterman, M.L.: Markov Decision Processes: Discrete Stochastic Dynamic Programming. Wiley-Interscience (1994)
22. Scherrer, B., Gabillon, V., Ghavamzadeh, M., Geist, M.: Approximate Modified Policy Iteration. In: International Conference on Machine Learning (ICML) (2012)
23. Scherrer, B., Lesner, B.: On the Use of Non-Stationary Policies for Stationary Infinite-Horizon Markov Decision Processes. In: Advances in Neural Information Processing Systems (NIPS) (2012)
24. Sutton, R., Barto, A.: Reinforcement Learning, An introduction. The MIT Press (1998)
25. Sutton, R.S., McAllester, D.A., Singh, S.P., Mansour, Y.: Policy Gradient Methods for Reinforcement Learning with Function Approximation. In: Advances in Neural Information Processing Systems (NIPS) (1999)