Dynamic Policy Programming with Function Approximation

Mohammad Gheshlaghi Azar
Department of Biophysics
Radboud University Nijmegen
Geert Grooteplein Noord 21
6525 EZ Nijmegen Netherlands
m.azar@science.ru.nl

Vicenç Gómez
Department of Biophysics
Radboud University Nijmegen
Geert Grooteplein Noord 21
6525 EZ Nijmegen Netherlands
v.gomez@science.ru.nl

Hilbert J. Kappen
Department of Biophysics
Radboud University Nijmegen
Geert Grooteplein Noord 21
6525 EZ Nijmegen Netherlands
b.kappen@science.ru.nl

Abstract

In this paper, we consider the problem of planning in the infinite-horizon discounted-reward Markov decision problems. We propose a novel iterative method, called dynamic policy programming (DPP), which updates the parametrized policy by a Bellman-like iteration. For discrete state-action case, we establish sup-norm loss bounds for the performance of the policy induced by DPP and prove that it asymptotically converges to the optimal policy. Then, we generalize our approach to large-scale (continuous) state-action problems using function approximation technique. We provide sup-norm performance-loss bounds for approximate DPP and compare these bounds with the standard results from approximate dynamic programming (ADP) showing that approximate DPP results in a tighter asymptotic bound than standard ADP methods. We also numerically compare the performance of DPP to other ADP and RL methods. We observe that approximate DPP asymptotically outperforms other methods on the mountain-car problem.

1 Introduction

Many problems in robotics, operations research and process control can be presented as **d**ynamic **p**rogramming (DP) problem. DP is based on estimation of some measures of the value of state (or state-action) through the Bellman equation. For high-dimensional systems or for continuous systems the state space is huge and computing the value function by DP is intractable. Common approaches to make the computation tractable are function-approximation approaches, where the value function is parameterized in terms of number of fixed basis functions and thus reduces the Bellman equation to the estimation of these parameters (Bertsekas and Tsitsiklis, 1996, chap. 6).

There are many algorithms for approximating the optimal value functions through DP (Szepesvari, 2009) and **a**pproximate **d**ynamic **p**rogramming (ADP) methods such as **a**pproximate **p**olicy **i**teration (API) and **a**pproximate **v**alue **i**teration (AVI) have been successfully applied to many real world problems. However, there are counter-examples in the literature where these methods fail to converge to a stable near-optimal solution (Bartlett, 2003; Bertsekas and Tsitsiklis, 1996, chap. 6). The main reason is that these algorithms switch the control policy without enforcing any smoothness in the policy. This lack of smoothness in the absence of accurate approximation of the value function, can drastically deteriorate the quality of the control policy since if the new policy is radically different than the previous one, it might be hard for the algorithm to recover from the failure in policy improvement. An incremental change of the policy may give a better chance to recover from failed updates. One of the most well-known algorithm of this kind is the **a**ctor-**c**ritic method (AC), in which the actor relies on the value function computed by the critic to guide the policy search (Sutton and Barto, 1998, chap. 6). An important extension of AC, the **p**olicy-**g**radient **a**ctor **c**ritic (PGAC) (Sutton et al., 1999; Peters and Schaal, 2008; Bhatnagar et al., 2009) updates the parameters of the policy in the direction of the (natural) gradient of performance estimated by the critic. The drawback of PGAC is that it suffers from local maxima since PGAC is basically a local search algorithm.

Appearing in Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS) 2011, Fort Lauderdale, FL, USA. Volume 15 of JMLR: W&CP 15. Copyright 2011 by the authors.

In this paper we introduce a new method to compute the optimal policy, called **d**ynamic **p**olicy **p**rogramming (DPP). DPP includes some of the features of AC. Like AC, DPP incrementally updates the parametrized policy. The difference is that DPP, instead of relying on value function for the policy update, uses the parameters of the policy to guide the policy search with a Bellman-like recursion.

The basic idea of DPP is to control the size of policy update by adding a term to the value function which penalizes large deviations from a baseline policy. By adding this penalty term, which is the relative entropy between some baseline policy and the control policy, we replace the maximization over actions in the right-hand side of the Bellman equation by a convex optimization problem. One can analytically solve this maximization problem and the solution for the control policy is directly expressed in terms of the value function, baseline policy and the specification of the environment. The value function itself is computed by a Bellman-like recursion. Iterating this process, where the new baseline policy becomes the just computed control policy, results in a double-loop iteration on the policy and the value function. DPP is the single-loop version of the preceding double-loop iteration, in which we combine the value iteration and the policy iteration in only one iteration on the action preferences. We then prove that the policy induced by the iterates of DPP asymptotically converges to the optimal policy. Further, we establish L_∞-loss bounds on the performance of the policy induced by DPP and generalize these bounds such that we can take the approximation error into account. We show that, for a given sequence of approximation errors, these bounds are tighter than previous bounds for ADP methods such as AVI and API. We also give an example for which the difference is dramatic.

This article is organized as follows. In section 2, we present the notations which are used in this paper. We introduce DPP and we investigate its convergence properties in section 3. In section 4, we demonstrate the compatibility of our method with approximation techniques. We also provide performance guarantee for DPP in the presence of approximation by generalizing the performance loss bounds of section 3. Section 5, presents numerical experiments on the mountain-car problem. In section 6 we briefly review the related works. Finally, we discuss some of the implications of our work in section 7.

2 Preliminaries

A stationary MDP is a 5-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, T, \gamma)$, where $\mathcal{S}, \mathcal{A}, \mathcal{R}$ are, respectively, the set of all system states, the set of actions that can be taken and the set of rewards that may be issued, such that $r^a_{ss'}$ denotes the reward of the next state s' given that the current state is s and the action is a. T is a set of matrices of dimension $|\mathcal{S} \times \mathcal{S}|$, one for each $a \in \mathcal{A}$ such that $T^a_{ss'}$ denotes the probability of the next state s' given the current state s and the action a. $\gamma \in (0, 1)$ denotes the discount factor.

Assumption 1. *We assume that for all 3-tuple $(s, a, s') \in \mathcal{S} \times \mathcal{A} \times \mathcal{S}$, the magnitude of the immediate reward, $|r^a_{ss'}|$ is bounded from above by R_{max}.*

A stationary policy is a mapping π that assigns to each state s a probability distribution over the action space $\mathcal{A}$, one for each $(s, a) \in \mathcal{S} \times \mathcal{A}$ such that $\pi_s(a)$ denotes the probability of the action a given the current state is s. Given the policy π, its corresponding value function V^π denotes the expected value of the long-term discounted sum of rewards in each state s, when the action is chosen by policy π. The goal is to find a policy π^* that attains the optimal value function $V^*(s)$, such that $V^*(s)$ satisfies a Bellman equation:

$$V^*(s) = \max_{\pi_s} \sum_{a \in \mathcal{A}} \pi_s(a) \sum_{s' \in \mathcal{S}} T^a_{ss'} \left(r^a_{ss'} + \gamma V^*(s') \right), \forall s \in \mathcal{S}. \quad (1)$$

Often it is convenient to associate values not with states but with state-action pairs. Therefore, we introduce the action-value functions: where $Q^\pi(s, a)$ denotes the expected value of the discounted sum of rewards for all $(s, a) \in \mathcal{S} \times \mathcal{A}$, when the future actions are chosen by the policy π. The optimal Q-function, Q^* satisfies a Bellman equation analogous to (1): $Q^*(s, a) = \sum_{s' \in \mathcal{S}} T^a_{ss'} (r^a_{ss'} + \gamma \max_{\pi_{s'}} \sum_{a' \in \mathcal{A}} \pi_{s'}(a') Q^*(s', a'))$.

3 Dynamic Policy Programming

In this section we derive DPP starting from the Bellman equation. We first show that by adding the relative entropy to the reward we can control the deviations of the optimal policy from a baseline policy. We then derive a double-loop approach which combines value and policy updates. We reduce this double-loop iteration to just a single iteration by introducing DPP algorithm. We emphasize that the purpose of the following *derivations* is to motivate DPP, rather than to provide a formal characterization. Subsequently, in section 3.2, we theoretically investigate the asymptotic behavior of DPP and prove its convergence.

3.1 From Bellman Equation to DPP Recursion

Consider the relative entropy between the policy π and some baseline policy $\bar{\pi}$:

$$g_{\bar{\pi}}^{\pi}(s) \triangleq \mathrm{KL}\left(\pi_s \| \bar{\pi}_s\right) = \sum_{a \in \mathcal{A}} \pi_s(a) \log\left(\frac{\pi_s(a)}{\bar{\pi}_s(a)}\right), \qquad \forall s \in \mathcal{S}.$$

We define a new value function $V_{\bar{\pi}}^{\pi}$ for all $s \in \mathcal{S}$ which incorporates g as a penalty term for deviating from the base policy $\bar{\pi}$ and the reward under the policy π:

$$V_{\bar{\pi}}^{\pi}(s) \triangleq \lim_{n\to\infty} \mathbb{E}_\pi \left[\sum_{k=1}^{n} \gamma^{k-1} \left(r_{s_{t+k}} - \frac{1}{\eta} g_{\bar{\pi}}^{\pi}(s_{t+k-1}) \right) \middle| s_t = s \right],$$

where η is a positive constant and $\mathbb{E}_\pi$ denotes expectation w.r.t. the state transition probability distribution T and the policy π. The optimal value function $V_{\bar{\pi}}^{*}(s) = \max_\pi V_{\bar{\pi}}^{\pi}(s)$ then satisfies the following Bellman equation for all $s \in \mathcal{S}$:

$$V_{\bar{\pi}}^{*}(s) = \max_{\pi_s} \sum_{\substack{a \in \mathcal{A} \\ s' \in \mathcal{S}}} \pi_s(a) \left[T_{ss'}^{a} \left(r_{ss'}^{a} + \gamma V_{\bar{\pi}}^{*}(s') \right) - \frac{1}{\eta} \log \frac{\pi_s(a)}{\bar{\pi}_s(a)} \right] \tag{2}$$

Equation (2) is a modified version of (1) where, in addition to maximizing the expected reward, the optimal policy $\bar{\pi}^*$ also minimizes the distance with the baseline policy $\bar{\pi}$. The maximization in (2) can be performed in closed form using Lagrange multipliers. Following Todorov (2006), we state lemma 1:

Lemma 1. *Let η be a positive constant, then for all $s \in \mathcal{S}$ the optimal value function $V_{\bar{\pi}}^{*}(s)$ and for all $(s,a) \in \mathcal{S} \times \mathcal{A}$ the optimal policy $\bar{\pi}_s^{*}(a)$, respectively, satisfy:*

$$V_{\bar{\pi}}^{*}(s) = \frac{1}{\eta} \log \sum_{a \in \mathcal{A}} \bar{\pi}_s(a) \exp\left[\eta \sum_{s' \in \mathcal{S}} T_{ss'}^{a} \left(r_{ss'}^{a} + \gamma V_{\bar{\pi}}^{*}(s') \right) \right] \tag{3}$$

$$\bar{\pi}_s^{*}(a) = \frac{\bar{\pi}_s(a) \exp\left[\eta \sum_{s' \in \mathcal{S}} T_{ss'}^{a} \left(r_{ss'}^{a} + \gamma V_{\bar{\pi}}^{*}(s') \right) \right]}{\exp\left(\eta V_{\bar{\pi}}^{*}(s) \right)} \tag{4}$$

Proof. See appendix B in supplementary material. □

The optimal policy $\bar{\pi}^*$ is a function of the base policy, the optimal value function $V_{\bar{\pi}}^{*}$ and the model data. One can first obtain the optimal value function $V_{\bar{\pi}}^{*}$ through the following fixed-point equation:

$$V_{\bar{\pi}}^{n+1}(s) = \frac{1}{\eta} \log \sum_{a \in \mathcal{A}} \bar{\pi}_s(a) \exp\left[\eta \sum_{s' \in \mathcal{S}} T_{ss'}^{a} \left(r_{ss'}^{a} + \gamma V_{\bar{\pi}}^{n} s' \right) \right], \tag{5}$$

and then compute a new policy $\bar{\pi}^*$ using (4). $\bar{\pi}^*$ maximizes the value function $V_{\bar{\pi}}^{\pi}$. However, we are not, in principle, interested in maximizing $V_{\bar{\pi}}^{\pi}$, but in maximizing the value function V^{π}. The idea to further improve the policy towards π^* is to replace the base policy with the just newly computed policy of (4). The new policy can be regarded as a *new base policy*, and the process can be repeated again. This leads to a double-loop algorithm to find the optimal policy π^*, where the outer-loop and the inner-loop would consist of a policy update, Equation (4), and a value function update, Equation (5), respectively.

Two more steps lead to the final DPP algorithm. First, note that one can replace the double-loop by direct optimization of both value function and policy simultaneously using the following fixed point iterations:

$$V_{\bar{\pi}}^{n+1}(s) = \frac{1}{\eta} \log \sum_{a \in \mathcal{A}} \bar{\pi}_s^{n}(a) \exp\left[\eta \sum_{s' \in \mathcal{S}} T_{ss'}^{a} \left(r_{ss'}^{a} + \gamma V_{\bar{\pi}}^{n}(s') \right) \right], \tag{6}$$

$$\bar{\pi}_s^{n+1}(a) = \frac{\bar{\pi}_s^{n}(a) \exp\left[\eta \sum_{s' \in \mathcal{S}} T_{ss'}^{a} \left(r_{ss'}^{a} + \gamma V_{\bar{\pi}}^{n}(s') \right) \right]}{\exp\left(\eta V_{\bar{\pi}}^{n+1}(s) \right)}. \tag{7}$$

Further, we can define action preferences (Sutton 1996) P_n for all $(s,a) \in \mathcal{S} \times \mathcal{A}$ and $n > 0$ as follows:

$$P_{n+1}(s,a) \triangleq \frac{1}{\eta} \log \bar{\pi}_s^{n}(a) + \sum_{s' \in \mathcal{S}} T_{ss'}^{a} \left(r_{ss'}^{a} + \gamma V_{\bar{\pi}}^{n}(s') \right). \tag{8}$$

By comparing (8) with (7) and (6), we deduce:

$$\bar{\pi}_s^{n}(a) = \frac{\exp(\eta P_n(s,a))}{\sum_{a' \in \mathcal{A}} \exp(\eta P_n(s,a'))}, \tag{9}$$

$$V_{\bar{\pi}}^{n}(s) = \frac{1}{\eta} \log \sum_{a \in \mathcal{A}} \exp(\eta P_n(s,a))). \tag{10}$$

Now by plugging (9) and (10) into (8) we derive:

$$P_{n+1}(s,a) = P_n(s,a) - \mathcal{L}_\eta P_n(s) + \sum_{s' \in \mathcal{S}} T_{ss'}^{a} \left(r_{ss'}^{a} + \gamma \mathcal{L}_\eta P_n(s') \right), \tag{11}$$

with the log-partition-sum operator $\mathcal{L}_\eta P_n(s) = 1/\eta \log \sum_{a' \in \mathcal{A}} \exp(\eta P(s,a'))$. (11) is one form of the DPP equations. There is a more efficient and analytically more tractable version of the DPP equation, where we replace the log-partion-sum $\mathcal{L}_\eta$ by the Boltzmann soft-max $\mathcal{M}_\eta$ defined by $\mathcal{M}_\eta P(s) = \sum_{a\in\mathcal{A}} \left[\exp(\eta P(s,a)) P(s,a) / \sum_{a'\in\mathcal{A}} \exp(\eta P(s,a'))\right]$ for all $(s,a) \in \mathcal{S}\times\mathcal{A}$.[1] In principle, we can provide formal analysis for both versions. However, the proof is somewhat simpler for the $\mathcal{M}_\eta$ case, which we will consider in the remainder of this paper. By replacing $\mathcal{L}_\eta$ with $\mathcal{M}_\eta$ we deduce the DPP recursion:

$$\begin{aligned} P_{n+1}(s,a) &= \mathcal{O}P_n(s,a) \\ &\triangleq P_n(s,a) - \mathcal{M}_\eta P_n(s) + \\ &\quad \sum_{s'\in\mathcal{S}} T^a_{ss'}\left(r^a_{ss'} + \gamma \mathcal{M}_\eta P_n(s')\right), \end{aligned} \tag{13}$$

with $\mathcal{O}$ is an operator defined on the action preferences P_n. Therefore, instead of iterating equations (6) and (7), the DPP algorithm updates action preferences via the DPP operator using Equation (13). In the next section we show that this iteration gradually moves the policy towards the greedy optimal policy. Algorithm 1 shows the procedure.

Algorithm 1: (DPP) Dynamic Policy Programming

```
Input: Randomized action preferences $P_0(.,.)$ and $\eta$
for $n = 1, 2, 3, \dots, N$ do
    for $(s,a) \in \mathcal{S}\times\mathcal{A}$ do
        $P_{n+1}(s,a) := P_n(s,a) - \mathcal{M}_\eta P_n(s) + \sum_{s'\in\mathcal{S}} T^a_{ss'}(r^a_{ss'} + \gamma\mathcal{M}_\eta P_n(s'))$;
    end
end
for $(s,a) \in \mathcal{S}\times\mathcal{A}$ do
    $\pi_s(a) := \frac{\exp(\eta P_N(s,a))}{\sum_{a'\in\mathcal{A}} \exp(\eta P_N(s,a'))}$;
end
return $\pi$;
```

3.2 Performance Guarantee

We provide the L_∞-norm performance-loss bounds for $Q^{\pi_n}(s,a)$, the action-value function of the policy induced by the nth iterate of DPP in theorem 1:

[1]Replacing $\mathcal{L}_\eta$ with $\mathcal{M}_\eta$ is motivated by the following relation between these two operators:

$$\mathcal{M}_\eta P(s) = \mathcal{L}_\eta P(s) + 1/\eta H_\pi(s), \quad \forall (s,a) \in \mathcal{S}\times\mathcal{A}, \tag{12}$$

with $H_\pi(s)$ is the entropy of the policy distribution π obtained by plugging P in to (9). For the proof of (12) and further readings see MacKay (2003, chap. 31).

Theorem 1. *Let assumption 1 hold. Also, for keeping the representation succinct, we assume that the magnitude of both $r^a_{ss'}$ and the initial action preferences $P_0(s,a)$ are bounded from above by some constant $L > 0$ for all $(s,a) \in \mathcal{S}\times\mathcal{A}$, then the following inequality holds:*

$$\max_{(s,a)\in\mathcal{S}\times\mathcal{A}} |Q^{\pi_n}(s,a) - Q^*(s,a)| \le \lambda_n,$$

where:

$$\lambda_n = 4\gamma \frac{(1-\gamma)^2 \log(|\mathcal{A}|)/\eta + 2L}{n(1-\gamma)^5} + 4\gamma^n \frac{L}{(1-\gamma)^2}$$

Proof. See appendix C in supplementary material. □

As an immediate corollary of theorem 1, we obtain the following result:

Corollary 1. *The following relation holds in limit:*

$$\lim_{n\to+\infty} Q^{\pi_n}(s,a) = Q^*(s,a), \quad \forall (s,a) \in \mathcal{S}\times\mathcal{A}.$$

In words the policy induced by DPP asymptotically converges to the optimal policy π^*. One can also show that, under some mild conditions, there exists a unique limit for the action preferences under DPP in infinity.

Assumption 2. *We assume that MDP has a unique deterministic optimal policy π^* given by:*

$$\pi^*_s(a) = \begin{cases} 1 & a = a^*(s) \\ 0 & \text{otherwise} \end{cases}, \qquad \forall s \in \mathcal{S},$$

where $a^*(s) = \arg\max_{a\in\mathcal{A}} Q^*(s,a)$.

Theorem 2. *Let assumption 2 holds and n be a positive integer and let $P_n(s,a)$ for all $(s,a) \in \mathcal{S}\times\mathcal{A}$ be the action preference after n iteration of DPP. Then, we have:*

$$\lim_{n\to+\infty} P_n(s,a) = \begin{cases} V^*(s) & a = a^*(s) \\ -\infty & \text{otherwise} \end{cases}, \quad \forall s \in \mathcal{S},$$

Proof. See appendix D in supplementary material. □

4 Dynamic Policy Programming with Approximation

Algorithm 1 (DPP) can only be applied to small problems with few states and actions. One can scale up DPP to large-scale state problems by using function approximation, where at each iteration n the action preferences P_n are the result of approximately applying the DPP operator, i.e., for all $(s,a) \in \mathcal{S}\times\mathcal{A}$: $P_{n+1}(s,a) \approx \mathcal{O}P_n(s,a)$. The approximation error ϵ_n is the difference of $\mathcal{O}P_n$ and its approximation:

$$\epsilon_n(s,a) \triangleq P_{n+1}(s,a) - \mathcal{O}P_n(s,a), \quad \forall (s,a) \in \mathcal{S}\times\mathcal{A} \tag{14}$$

In this section we provide results on the performance-loss of DPP in the presence of approximation error. We then compare L_∞-norm performance-loss bounds of DPP with the standard results of approximate value and policy iteration. Finally, we introduce an algorithm for implementing approximate DPP with linear function approximation.

4.1 L_∞-Norm Performance-Loss Bound for Approximate DPP

The following theorem establish an upper-bound for L_∞-norm of performance loss of DPP in the presence of approximation error. The proof is based on generalization of the bound that we established for DPP such that it take care of the approximation error:

Theorem 3 (L_∞ performance loss bound of approximate DPP)**.** *Let assumption 1 hold, $|\mathcal{A}|$ be the cardinality of the action space $\mathcal{A}$. Also define ϵ_n according to* (14)*. Further, for keeping the representation succinct, assume that $r^a_{ss'}$, P_0 and ϵ_k (for all $k \geq 1$) are uniformly bounded by some constant $L > 0$ and for any positive integer n, and the L_∞-norm of average error at iteration n, $\bar{\varepsilon}_n$, is defined as follows:*

$$\bar{\varepsilon}_n \triangleq \max_{(s,a)\in\mathcal{S}\times\mathcal{A}} \left| \frac{1}{n}\sum_{k=0}^{n-1} \epsilon_k(s,a) \right| \tag{15}$$

Then the following inequality holds for the policy induced by approximate DPP at iteration n:

$$\max_{(s,a)\in\mathcal{S}\times\mathcal{A}} |Q^{\pi_n}(s,a) - Q^*(s,a)| \leq \lambda_n,$$

where:

$$\lambda_n = 4\gamma \frac{(1-\gamma)^2 \log(|\mathcal{A}|)/\eta + 4L}{n(1-\gamma)^5} + 4\gamma^n \frac{L}{(1-\gamma)^2} + \frac{2\gamma}{1-\gamma}\sum_{k=1}^{n} \gamma^{n-k}\bar{\varepsilon}_k$$

Proof. See appendix E in supplementary material. □

Taking the upper-limit yields in the following corollary of theorem 3.

Corollary 2 (Asymptotic L_∞-norm performance-loss bound of approximate DPP)**.** *Assume that $\bar{\varepsilon}_n$ is defined from* (15)*. Then, the following inequality holds:*

$$\limsup_{n\to\infty} \max_{(s,a)\in\mathcal{S}\times\mathcal{A}} |Q^{\pi_n}(s,a) - Q^*(s,a)| \leq \frac{2\gamma}{(1-\gamma)^2}\bar{\varepsilon} \tag{16}$$

where $\bar{\varepsilon} = \limsup_{n\to\infty}\bar{\varepsilon}_n$.

This bound is quite similar to the bound derived by Bertsekas and Tsitsiklis (1996) for AVI and those of the optimistic approximate policy iteration (OAPI) (Thiery and Scherrer, 2010; Bertsekas and Tsitsiklis, 1996, chap. 6):

$$\limsup_{n\to\infty} \max_{(s,a)\in\mathcal{S}\times\mathcal{A}} |Q^{\pi_n}(s,a) - Q^*(s,a)| \leq \frac{2\gamma}{(1-\gamma)^2}\varepsilon_{\max},$$

with $\varepsilon_{\max} = \max_{n\geq 0}\max_{(s,a)\in\mathcal{S}\times\mathcal{A}} |\epsilon_n(s,a)|$. The difference is that in (16) the sup-norm error $\varepsilon_{\max}$ is replaced by asymptotic sup-norm of average error $\bar{\varepsilon}$ for which one can easily show from generalized Phytagorean theorem that $\bar{\varepsilon} \leq \varepsilon_{\max}$ for a given infinite sequence of approximation error $\{\epsilon_1, \epsilon_2, \dots\}$. Therefore, ADPP can provide a tighter upper bound on asymptotic performance than approximate value iteration and optimistic policy iteration. To have a better understanding of difference between these two bounds we consider the following simple example:

Example 1. *Consider a problem in which the sequence of approximation error $\{\epsilon_{1:n}\}$ are i.i.d. samples of a bounded zero-mean random variable ϵ. Then we obtain the following asymptotic bounds for approximate DPP, OAPI and AVI:*

Approximate DPP:

$$\limsup_{n\to\infty} \max_{\substack{s\in\mathcal{S}\\ a\in\mathcal{A}}} |Q^{\pi_n}(s,a) - Q^*(s,a)| \leq \frac{2\gamma}{(1-\gamma)^2}\bar{\varepsilon} = 0.$$

OAPI and AVI:

$$\limsup_{n\to\infty} \max_{\substack{s\in\mathcal{S}\\ a\in\mathcal{A}}} |Q^{\pi_n}(s,a) - Q^*(s,a)| \leq \frac{2\gamma}{(1-\gamma)^2}\varepsilon_{\max}.$$

In words, the bounds suggest that approximate DPP asymptotically manages to cancel the i.i.d. noise and converges to the optimal policy, whereas there is no guarantee for the convergence of OAPI and AVI to the optimal solution in this case.

The fact that the L_∞-norm bounds of DPP are expressed in terms of the sup-norm of average error instead of sup-norm of error itself may be useful in estimating DPP operator by Monte-Carlo sampling. Each step of DPP consists of an expected value over the action preferences of the next state and the next action. In large-scale problems, exact evaluation of this expectation is not feasible. The alternative is to use Monte-Carlo simulation to estimate DPP operator. However, by estimating DPP operator through Monte-Carlo simulation we introduce a simulation noise term (estimation error) to the problem. The hope is that DPP, in the light of what we observe in example 1, asymptotically cancels the estimation error caused by Monte-Carlo simulation and provides a better estimation of the optimal control than sampling-based ADP methods which may propagate the estimation error (Munos and Szepesvári, 2008).

4.2 Approximate Dynamic Policy Programming with Linear Function Approximation

In this subsection, we provide a solution for approximating DPP operator using linear function approximation and the least-squares regression. We also introduce the ADPP algorithm based on this solution.

Before we proceed with the idea of approximating DPP operator, we find it convenient to re-express some of our results in section 3 in vector notation. $\mathbf{p}$ denotes a $|\mathcal{S}||\mathcal{A}|\times 1$ vector of the action preferences with $\mathbf{p}(sa)=P(s,a)$ and $P(s,a)$ relates to the policy by (9). Furthermore, $\boldsymbol{\mathcal{O}}\mathbf{p}$ denotes a $|\mathcal{S}||\mathcal{A}|\times 1$ vector, such that, $\boldsymbol{\mathcal{O}}\mathbf{p}(sa)=\mathcal{O}P(s,a)$ for all $(s,a)\in\mathcal{S}\times\mathcal{A}$. Let us define $\mathbf{\Phi}$ as a $|\mathcal{S}||\mathcal{A}|\times k$ matrix, where for each $(s,a)\in\mathcal{S}\times\mathcal{A}$, the row vector $\mathbf{\Phi}(sa,\cdot)$ denotes the output of set of basis functions $\mathcal{F}_\phi=[\phi_1,\cdots,\phi_k]$ given the state-action pair (s,a) as the input, where each basis function $\phi_i:\mathcal{S}\times\mathcal{A}\longrightarrow\Re$.

A parametric approximation for the action preferences $\hat{\mathbf{p}}$ can then be expressed as a projection of the action preferences onto the column space spanned by $\mathbf{\Phi}$, $\hat{\mathbf{p}}=\mathbf{\Phi}\boldsymbol{\theta}$, where $\boldsymbol{\theta}\in\Re^k$ is a $k\times 1$ vector of parameters.

There is no guarantee that $\boldsymbol{\mathcal{O}}\hat{\mathbf{p}}$ stays in the column span of $\mathbf{\Phi}$. Instead, a common approach is to find a vector $\boldsymbol{\theta}$ that projects $\boldsymbol{\mathcal{O}}\hat{\mathbf{p}}$ on the column space spanned by $\mathbf{\Phi}$, while minimizing some pseudo-normed error $J\triangleq\|\boldsymbol{\mathcal{O}}\hat{\mathbf{p}}-\mathbf{\Phi}\boldsymbol{\theta}\|^2_{\mathbf{\Upsilon}}=\big(\boldsymbol{\mathcal{O}}\hat{\mathbf{p}}-\mathbf{\Phi}\boldsymbol{\theta}\big)^{\mathsf{T}}\mathbf{\Upsilon}\big(\boldsymbol{\mathcal{O}}\hat{\mathbf{p}}-\mathbf{\Phi}\boldsymbol{\theta}\big)$, where $\mathbf{\Upsilon}$ is a $|\mathcal{S}||\mathcal{A}|\times|\mathcal{S}||\mathcal{A}|$ diagonal matrix and $\sum_{sa}\mathrm{diag}\big(\mathbf{\Upsilon}\big)(sa)=1$ (Bertsekas, 2007, chap. 6).[2] The best solution, that minimize J, is given by the least-squares projection:

$$\boldsymbol{\theta}^*=\arg\min_{\boldsymbol{\theta}\in\Re^k}J(\boldsymbol{\theta})=\big(\mathbf{\Phi}^{\mathsf{T}}\mathbf{\Upsilon}\mathbf{\Phi}\big)^{-1}\mathbf{\Phi}^{\mathsf{T}}\mathbf{\Upsilon}\boldsymbol{\mathcal{O}}\hat{\mathbf{p}}. \tag{17}$$

Equation (17) requires the computation of $\boldsymbol{\mathcal{O}}\hat{\mathbf{p}}$ for all states and actions. For large scale problems this becomes infeasible. However, we can simulate the state trajectory and then make an empirical estimate of the least-squares projection. The key observation in deriving a sample estimate of (17) is that the pseudo-normed error J can be written as an expectation:

$$J=\mathbb{E}_{sa\sim\mathbf{\Upsilon}}\big[(\boldsymbol{\mathcal{O}}\hat{\mathbf{p}}(sa)-\mathbf{\Phi}\boldsymbol{\theta}(sa))^{\mathsf{T}}(\boldsymbol{\mathcal{O}}\hat{\mathbf{p}}(sa)-\mathbf{\Phi}\boldsymbol{\theta}(sa))\big].$$

Therefore, one can simulate the state trajectory according to $\boldsymbol{\pi}$ (to be specified) and then empirically estimate J and the least-squares solution for the corresponding visit distribution matrix $\mathbf{\Upsilon}$.

In an analogy with Bertsekas (2007, chap. 6.1), we can derive a sample estimate of (17). We assume that a trajectory of $m+1$ ($m\gg 1$) state-action pairs, $\mathcal{G}_{\boldsymbol{\pi}}=\{(s_1,a_1),(s_2,a_2)\cdots,(s_{m+1},a_{m+1})\}$, is available. The trajectory $\mathcal{G}_{\boldsymbol{\pi}}$ is generated by simulating the policy $\boldsymbol{\pi}$ for $m+1$ steps and has a stationary distribution $\mathbf{\Upsilon}$. Then, we construct the $m\times k$ matrix $\tilde{\mathbf{\Phi}}$ with $\tilde{\mathbf{\Phi}}(i,j)=\phi_j(s_i,a_i)$, where (s_i,a_i) is the i^{th} component of $\mathcal{G}_{\boldsymbol{\pi}}$. An unbiased estimation of (17) under the stationary distribution $\mathbf{\Upsilon}$ can then be identified as follows:

$$\tilde{\boldsymbol{\theta}}^*=\big(\tilde{\mathbf{\Phi}}^{\mathsf{T}}\tilde{\mathbf{\Phi}}\big)^{-1}\tilde{\mathbf{\Phi}}^{\mathsf{T}}\hat{\tilde{\mathbf{p}}}, \tag{18}$$

where $\hat{\tilde{\mathbf{p}}}$ is a $m\times 1$ vector with entries:

$$\begin{aligned}\hat{\tilde{\mathbf{p}}}(i)\triangleq{}&\hat{\mathbf{p}}(s_ia_i)+\mathbf{r}(s_ia_i,s_{i+1})\\&+\gamma\boldsymbol{\mathcal{M}}_\eta\hat{\mathbf{p}}(s_{i+1})-\boldsymbol{\mathcal{M}}_\eta\hat{\mathbf{p}}(s_i),\quad i=1,2,\cdots,m.\end{aligned} \tag{19}$$

By comparing (19) with (13), we observe that $\boldsymbol{\mathcal{O}}\mathbf{p}(s_ia_i)=\mathbb{E}_{s_{i+1}\sim\mathbf{T}}\big(\hat{\tilde{\mathbf{p}}}(i)\big)$.

So far, we have not specified the policy $\boldsymbol{\pi}$ that is used to generate $\mathcal{G}_{\boldsymbol{\pi}}$. There are several possibilities (Sutton and Barto, 1998, chap. 5). Here, we choose the on-policy approach, where the state-action trajectory $\mathcal{G}_{\boldsymbol{\pi}}$ is generated by the policy induced by $\hat{\mathbf{p}}$. Algorithm 2 presents on-policy ADPP method which relies on (18) to approximate DPP operator at each iteration.

Algorithm 2: (ADPP) On-policy approximate dynamic policy programming

```
Input: Randomized parameterized $\hat{\mathbf{p}}(\boldsymbol{\theta})$, $\eta$ and $m$
for $n=1,2,3,\ldots$ do
    Construct the policy $\hat{\boldsymbol{\pi}}$ from $\hat{\mathbf{p}}(\boldsymbol{\theta})$ ;
    Make a $m+1$ sequence $\mathcal{G}$ of $(s,a)\in\mathcal{S}\times\mathcal{A}$ by simulating $\hat{\boldsymbol{\pi}}$;
    Construct $\hat{\tilde{\mathbf{p}}}$ and $\tilde{\mathbf{\Phi}}$ from the state trajectory $\mathcal{G}$;
    $\tilde{\boldsymbol{\theta}}^*=\big(\tilde{\mathbf{\Phi}}^{\mathsf{T}}\tilde{\mathbf{\Phi}}\big)^{-1}\tilde{\mathbf{\Phi}}^{\mathsf{T}}\hat{\tilde{\mathbf{p}}}$;
    $\boldsymbol{\theta}\leftarrow\tilde{\boldsymbol{\theta}}^*$;
end
return $\boldsymbol{\theta}$
```

In order to estimate the optimal policy by ADPP we only need some trajectories of state-action-reward generated by Monte-Carlo simulation (i.e., knowledge of the transition probabilities is not required). In other words, ADPP directly *learns* an approximation of the optimal policy by Monte-Carlo simulation.

5 Numerical Results

In this section, we investigate the effectiveness of ADPP on the well-known mountain-car problem and compare its performance with standard approximate dynamic programming and reinforcement learning methods. The mountain-car domain has been reported to diverge in the presence of approximation (Boyan and Moore, 1995), although successful results have been observed by carefully crafting the basis functions

[2]One can associate diag($\mathbf{\Upsilon}$) with the stationary distribution of state-action pairs (s,a) for some policy $\boldsymbol{\pi}$ and the transition matrix $\mathbf{T}$.

(Sutton, 1996). The specification of the mountain-car domain is described by Sutton and Barto (1998, chap. 8).

We compare the performance of algorithm 2 (ADPP) with the **a**pproximate **Q**-**i**teration (AQI) (Bertsekas, 2007, chap. 6) and OAPI on the mountain-car problem.[3] We also report results on the natural actor critic, NAC-LSTD, (Peters and Schaal, 2008). The step size β_n is defined for the actor update rule of NAC-LSTD:

$$\beta_n \triangleq \frac{\beta_0}{t_\beta n + 1}, \qquad n = 1, 2, 3, \dots,$$

with the free parameters β_0 and t_β are some positive constants. Also, the temperature factor τ_n for the soft-max in OAPI with soft-max policy is given by:

$$\tau_n \triangleq \frac{\tau_0}{t_\tau \log(n+1) + 1}, \qquad n = 1, 2, 3, \dots,$$

where the free parameters τ_0 and t_τ are some positive constants. All the free parameters are optimized for the best asymptotic performance. Also, for each trial, we initialize all the algorithms in a random fashion.

As a measure of performance evaluation, we use the root mean-squared error (RMSE) between the value functions $\mathbf{v}^{\hat{\boldsymbol{\pi}}_n}$ under the policy induced by the corresponding algorithm at iteration n and the optimal value functions $\mathbf{v}^*$: RMSE $\triangleq \left\| \mathbf{v}^{\hat{\boldsymbol{\pi}}_n} - \mathbf{v}^* \right\|_2 \big/ \sqrt{|\mathcal{S}|}$.

To obtain an accurate estimate of the optimal value function, we discretize the state space with a 1751×151 grid and solve the resulting infinite-horizon MDP using value iteration (Bertsekas, 2007, chap. 1). The resulting value function is just used to derive the RMSE for the approximate algorithms and computing the optimal policy for discretized MDP.

In addition to standard approximate-DP algorithms, we compare our results with the performance of the best linear function approximator (optimal FA). The best linear function approximator is specified as the projection of the optimal policy π^* of discretized MDP onto the column space spanned by $\mathbf{\Phi}$ defined in section 4. Here, we consider $k = 27$ random radial basis functions to approximate the state-action dependent quantities, i.e., the action-value functions, the action preferences (NAC-LSTD and ADPP) and the approximate optimal policy $\hat{\boldsymbol{\pi}}^*$ (the optimal FA), and $k = 9$ radial basis functions to approximate the state-dependent value functions in NAC-LSTD. we also fix the number of samples m to 5×10^5 which we refresh then at each iteration.[4] Figure 1. a shows the RMSE of all algorithms as a function of the number of iterations. First, we can see that ADPP asymptotically outperforms the other methods and substantially narrows the gap with the optimal FA performance. ADPP is therefore asymptotically the best approximate algorithm which can be applied to large instances.

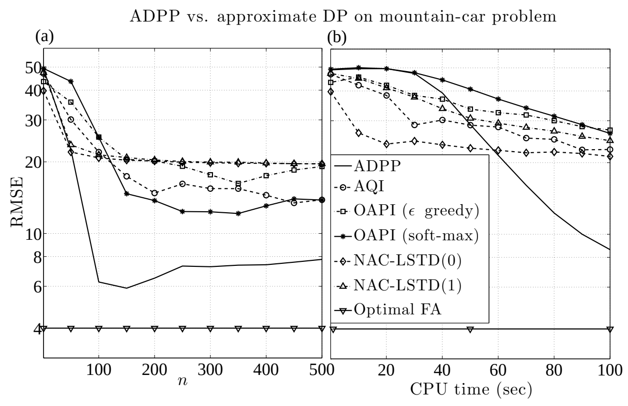

Figure 1: RMSE in terms of (a) number of iterations and (b) CPU time. The results are averages over 50 runs.

We also report a small decline in performance of both ADPP and OAPI (soft-max) after the corresponding RMSEs are minimized at iteration 150 and iteration 350. This is a minor detail and can be explained by the fact that both ADPP and OAPI are on-policy algorithms. Such approaches tend to increase the number of samples for the high-value states and reduce the number of samples for the low-value ones. This specialization leads to an eventual small increase of the RMSE measure, which does not differentiate between states that are more frequent than others.[5]

To compare the transient behavior of the algorithms, we plot the RMSE in terms of the computational cost (CPU time) for the first 100 seconds of simulation (Figure 1. b). We observe that, while some DP methods, in particular NAC-LSTD(0), improves fast in the early stages of optimization, ADPP performs better in the long term (one minute in this case). Overall, we conclude that DPP, when combined with the function approximation, reaches a near-optimal performance and

[3]In addition to ϵ-greedy policy ($\epsilon = 0.01$), we also implement OAPI with a soft-max policy, since the existence of fixed point is guaranteed in this case (Farias and Roy, 2000). Further, to facilitate the computation of the control policy, we approximate the action-value functions as opposed to the value functions in OAPI.

[4]These basis functions are randomly placed within the range of their inputs. The variance matrix is identical for all the basis functions:

$$\mathbf{\Sigma} \triangleq \begin{pmatrix} \sigma_x^2 & 0 & 0 \\ 0 & \sigma_{\dot{x}}^2 & 0 \\ 0 & 0 & \sigma_u^2 \end{pmatrix} = \begin{pmatrix} 0.1606 & 0 & 0 \\ 0 & 0.0011 & 0 \\ 0 & 0 & 0.2222 \end{pmatrix},$$

[5]The RMSE of ADPP at iteration $n = 5000$ was 8.2.

outperforms the methods considered here. ADPP can be slow in the early stages of the optimization process, but improves by a wide margin in the long term.

6 Related Work

There are some other approaches based on iterating the parametrized policy. The most popular is a gradient-based search for the optimal policy, which directly estimates the gradient of the performance with respect to the parameters of the control policy by Monte-Carlo simulation (Kakade, 2001; Baxter and Bartlett, 2001). Although the gradient-based policy-search methods guarantee convergence to a local maximum, they suffer from high variance and local maxima.

Wang et al. (2007) introduce a dual representation technique called dual dynamic programming (dual-DP) based on manipulating the state-action visit distributions . They have reported better convergence results than value-based methods in the presence of function approximation. The drawback of the dual approach is that, it needs more memory space than the primal representation.

The work proposed in this paper has some relation to recent work by Kappen (2005); Todorov (2006), who formulate the control cost as a relative entropy between the controlled and uncontrolled dynamics. The difference with DPP is that in their work a restricted class of control problems is considered. Instead, the present approach is more general.

Another relevant study is **r**elative **e**ntropy **p**olicy **s**earch (REPS) (Peters et al., 2010) which relies on the idea of minimizing the relative entropy to control the size of policy update. The main differences are: 1) the REPS algorithm is an actor-critic type of algorithm (without using a gradient in the actor), while DPP is more a policy iteration type of method, 2) In REPS η is also optimized while here is fixed, and 3) here we provide a convergence analysis of DPP, while there is no convergence analysis in REPS.

7 Discussion and Future Works

We have presented a novel policy-search method, dynamic policy programming (DPP), to compute the optimal policy through DPP operator. We have proven the convergence of our method to the optimal policy theoretically for the tabular case. We also have provided a L_∞-norm performance-loss bounds for DPP and generalize these bounds such that it handles approximation. The L_∞-norm loss-bounds suggest that DPP can perform better than ADP methods in the presence of approximation error. Experimental results for the mountain car problem confirms our theoretical results and show that DPP asymptotically outperforms the standard ADP and RL methods. On the other hand, ADPP makes little progress towards the optimal performance at the early stages of optimization. This behavior is also predicted by our performance loss bounds, since DPP loss-bound decays with linear rate as opposed to greedy ADP methods in which the loss-bounds decay with an exponential rate. NAC-LSTD(0) is also faster than DPP, since it moves the policy in the direction of the gradient and converges to a locally optimal solution very fast but its asymptotic performance is inferior to DPP.

In this study, we provide L_∞-norm performance-loss bounds for approximate DPP. However, most supervised learning and regression algorithms rely on minimizing some form of L_p-norm error. Therefore, it is natural to search for a kind of performance bounds that relies on the L_p-norm of approximation error. Following Munos (2005), L_p-norm bounds for approximate DPP can be established by providing a bound on the performance loss of each component of value function under the policy induced by DPP. This is a part of ongoing research which will be published elsewhere.

Another direction for future work is to provide finite-sample performance-loss bounds for the sampling-based approximate DPP in the spirit of previous theoretical results available for fitted value iteration and fitted Q-iteration by Antos et al. (2008); Munos and Szepesvári (2008).

Finally, an important extension of our results would be to apply DPP for large-scale action problems. In that case, we need an efficient way to approximate $\mathcal{M}_\eta P(s)$ in update rule (13) since computing the exact summations become expensive. One idea is to sample estimate $\mathcal{M}_\eta P(s)$ using Monte-Carlo simulation (MacKay, 2003, chap. 29), since $\mathcal{M}_\eta P(s)$ is the expected value of $P(s,a)$ under the soft-max policy π.

References

Antos, A., Munos, R., and Szepesvári, C. (2008). Fitted q-iteration in continuous action-space mdps. In *Proceedings of the 21st Annual Conference on Neural Information Processing Systems.*

Bartlett, P. L. (2003). An introduction to reinforcement learning theory: Value function methods. *Lecture Notes in Artificial Intelligence*, 2600/2003:184–202.

Baxter, J. and Bartlett, P. L. (2001). Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, 15:319–350.

Bertsekas, D. P. (2007). *Dynamic Programming and*

Optimal Control, volume II. Athena Scientific, Belmount, Massachusetts, third edition.

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

Bhatnagar, S., Sutton, R. S., Ghavamzadeh, M., and Lee, M. (2009). Natural actor-critic algorithms. *Automatica*, 45(11):2471–2482.

Boyan, J. A. and Moore, A. W. (1995). Generalization in reinforcement learning: Safely approximating the value function. In *Advances in Neural Information Processing Systems*, pages 369–376.

Farias, D. P. and Roy, B. V. (2000). On the existence of fixed points for approximate value iteration and temporal-difference learning. *Journal of Optimization Theory and Applications*, 105(3):589–608.

Kakade, S. (2001). Natural policy gradient. In *Advances in Neural Information Processing Systems 14*, pages 1531–1538, Vancouver, British Columbia, Canada.

Kappen, H. J. (2005). Path integrals and symmetry breaking for optimal control theory. *Statistical Mechanics*, 2005(11):P11011.

MacKay, D. J. C. (2003). *Information Theory, Inference, and Learning Algorithms.* Cambridge University Press, Cambridge, United Kingdom, first edition.

Munos, R. (2005). Error bounds for approximate value iteration. In *Proceedings of the 20th National Conference on Artificial Intelligence*, volume II, pages 1006–1011, Pittsburgh, Pennsylvania.

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

Peters, J., Mülling, K., and Altun, Y. (2010). Relative entropy policy search. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, Atlanta, Georgia, USA.*

Peters, J. and Schaal, S. (2008). Natural actor-critic. *Neurocomputing*, 71(7–9):1180–1190.

Sutton, R. S. (1996). Generalization in reinforcement learning: succesful examples using sparse coarse coding. In *Advances in Neural Information Processing Systems 9*, pages 1038–1044.

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

Sutton, R. S., McAllester, D., Singh, S., and Mansour, Y. (1999). Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems 12*, pages 1057–1063, Denver, Colorado, USA.

Szepesvari, C. (2009). Reinforcement learning algorithms for mdps – a survey. Technical Report TR09–13, Department of Computing Science, University of Alberta, Edmonton, Alberta, Canada.

Thiery, C. and Scherrer, B. (2010). Least-squares policy iteration: Bias- variance trade-off in control problems. In *Proceedings of the 27th Annual International Conference on Machine Learning.*

Todorov, E. (2006). Linearly-solvable markov decision problems. In *Proceedings of the 20th Annual Conference on Neural Information Processing Systems*, pages 1369–1376, Vancouver, British Columbia, Canada.

Wang, T., Lizotte, D., Bowling, M., and Schuurmans, D. (2007). Stable dual dynamic programming. In *Proceedings of the 21st Annual Conference on Neural Information Processing Systems*, Vancouver, British Columbia, Canada.