Approximate Dynamic Programming By Minimizing Distributionally Robust Bounds

Marek Petrik MPETRIK@US.IBM.COM
IBM T.J. Watson Research Center, Yorktown, NY, USA

Abstract

Approximate dynamic programming is a popular method for solving large Markov decision processes. This paper describes a new class of approximate dynamic programming (ADP) methods—distributionally robust ADP—that address the curse of dimensionality by minimizing a pessimistic bound on the policy loss. This approach turns ADP into an optimization problem, for which we derive new mathematical program formulations and analyze its properties. DRADP improves on the theoretical guarantees of existing ADP methods—it guarantees convergence and L_1 norm-based error bounds. The empirical evaluation of DRADP shows that the theoretical guarantees translate well into good performance on benchmark problems.

1. Introduction

Large Markov decision processes (MDPs) are common in reinforcement learning and operations research and are often solved by approximate dynamic programming (ADP). Many ADP algorithms have been developed and studied, often with impressive empirical performance. However, because many ADP methods must be carefully tuned to work well and offer insufficient theoretical guarantees, it is important to develop new methods that have both good theoretical guarantees and empirical performance.

Approximate linear programming (ALP)—an ADP method—has been developed with the goal of achieving convergence and good theoretical guarantees (de Farias & van Roy, 2003). Approximate bilinear programming (ABP) improves on the theoretical properties of ALP at the cost of additional computational complexity (Petrik & Zilberstein, 2009; 2011). Both ALP and ABP provide guarantees that rely on conservative error bounds in terms of the L_∞ norm and often under-perform in practice (Petrik & Zilberstein, 2009). It is, therefore, desirable to develop ADP methods that offer both tighter bounds and better empirical performance.

In this paper, we propose and analyze *distributionally robust approximate dynamic programming* (DRADP)—a new approximate dynamic programming method. DRADP improves on approximate linear and bilinear programming both in terms of theoretical properties and empirical performance. This method builds on approximate linear and bilinear programming but achieves better solution quality by explicitly optimizing tighter, less conservative, bounds stated in terms of a weighted L_1 norm. In particular, DRADP computes a good solution for a given initial distribution instead of attempting to find a solution that is good for all initial distributions.

The objective in ADP is to compute a policy π with the maximal return $\rho(\pi)$. Maximizing the return also minimizes the loss with respect to the optimal policy $\pi^\star$—known as the *policy loss* and defined as $\rho(\pi^\star) - \rho(\pi)$. There are two main challenges in computing a good policy for a large MDP. First, it is necessary to efficiently evaluate its return; evaluation using simulation is time consuming and often impractical. Second, the return of a parameterized policy may be a function that is hard to optimize. DRADP addressed both these issues by maximizing a simple *lower bound* $\tilde{\rho}(\pi)$ on the return using ideas from robust optimization. This lower bound is easy to optimize and can be computed from a small sample of the domain, eliminating the need for extensive simulation.

Maximizing a lower bound on the return corresponds to minimizing an upper bound $\rho(\pi^\star) - \tilde{\rho}(\pi)$ on the policy loss. The main reason to minimize an upper bound—as opposed to a lower bound—is that the approximation error can be bounded by the difference

Appearing in *Proceedings of the 29th International Conference on Machine Learning*, Edinburgh, Scotland, UK, 2012.

$\rho(\pi^\star) - \tilde{\rho}(\pi^\star)$ for the optimal policy only, instead of the difference for the set of all policies, as we show formally in Section 4.

The lower bound on the return in DRADP is based on an approximation of the *state occupancy distributions* or frequencies (Puterman, 2005). The state occupancy frequency represents the fraction of time that is spent in the state and is in some sense the dual of a value function. Occupancy frequencies have been used, for example, to solve factored MDPs (Dolgov & Durfee, 2006) and in dual dynamic programming (Wang, 2007; Wang et al., 2008) (The term "dual dynamic programming" also refers to unrelated linear stochastic programming methods). These methods can improve the empirical performance, but proving bounds on the policy loss has proved challenging. We take a different approach to prove tight bounds on the policy loss. While the existing methods approximate the state occupancy frequencies by a *subset*, we approximate it by a *superset*.

We call the DRADP approach distributionally robust because it uses the robust optimization methodology to represent and simplify the set of occupancy distributions (Delage & Ye, 2010). *Robust optimization* is a recently revived approach for modeling uncertainty in optimization problems (Ben-Tal et al., 2009). It does not attempt to model the uncertainty precisely, but instead computes solutions that are immunized against its effects. In distributionally robust optimization, the uncertainty is in probability distributions. DRADP introduces the uncertainty in state occupancy frequencies in order to make very large MDPs tractable and uses the robust optimization approach to compute solutions that are immune to this uncertainty.

The remainder of the paper is organized as follows. First, in Section 2, we define the basic framework including MDPs and value functions. Then, Section 3 introduces the general DRADP method in terms of generic optimization problems. Section 4 analyzes approximation errors involved in DRADP and shows that standard concentration coefficient assumptions on the MDP (Munos, 2007) can be used to derive tighter bounds. To leverage existing mathematical programming methods, we show that DRADP can be formulated in terms of standard mathematical optimization models in Section 5. Finally, Section 6 presents experimental results on standard benchmark problems. Due to space constraints we omit the proofs in this version; please see the extended version for the full proofs (Petrik, 2012).

We consider the offline—or batch—setting in this paper, in which all samples are generated in advance of computing the value function. This setting is identical to that of LSPI (Lagoudakis & Parr, 2003) and ALP (de Farias & van Roy, 2003).

2. Framework and Notation

In this section, we define the basic concepts required for solving Markov decision processes: value functions, and occupancy frequencies. We use the following general notation throughout the paper. The symbols $\mathbf{0}$ and $\mathbf{1}$ denote vectors of all zeros or ones of appropriate dimensions respectively; the symbol $\mathbf{I}$ denotes an identity matrix of an appropriate dimension. The operator $[\cdot]_+$ denotes an element-wise non-negative part of a vector. We will often use linear algebra and expectation notations interchangeably; for example: $\mathbf{E}_u[X] = u^\mathsf{T} x$, where x is a vector of the values of the random variable X. We also use $\mathbb{R}^\mathcal{X}$ to denote the set of all functions from a finite set $\mathcal{X}$ to $\mathbb{R}$; note that $\mathbb{R}^\mathcal{X}$ is trivially a vector space.

A *Markov Decision Process* is a tuple $(\mathcal{S}, \mathcal{A}, P, r, \alpha)$. Here, $\mathcal{S}$ is a *finite* set of states, $\mathcal{A}$ is a *finite* set of actions, $P : \mathcal{S}\times\mathcal{A}\times\mathcal{S} \mapsto [0,1]$ is the transition function ($P(s,a,s')$ is the probability of transiting to state s' from state s given action a), and $r : \mathcal{S}\times\mathcal{A} \mapsto \mathbb{R}$ is a reward function. The initial distribution is: $\alpha : \mathcal{S} \mapsto [0,1]$, such that $\sum_{s\in\mathcal{S}} \alpha(s) = 1$. The set of all state-action pairs is $\mathcal{W} = \mathcal{S}\times\mathcal{A}$. For the sake of simplicity, we assume that all actions can be taken in all states. To avoid technicalities that detract from the main ideas of the paper, we assume finite state and action sets but the results apply with additional compactness assumptions to infinite sets. We will use S and W to denote random variables with values in $\mathcal{S}$ and $\mathcal{W}$.

The solution of an MDP is a stationary *deterministic policy* $\pi : \mathcal{S} \to \mathcal{A}$, which determines the action to take in any state; the set of all deterministic policies is denoted by Π_D. A stationary *randomized—or stochastic—policy* $\pi : \mathcal{S}\times\mathcal{A} \to [0,1]$ assigns the probability to all actions for every state; the set of all randomized policies is denoted by Π_R. Clearly $\Pi_D \subseteq \Pi_R$ holds by mapping the chosen action to the appropriate distribution. A randomized policy can be thought of as a vector on $\mathcal{W}$ that assigns the appropriate probabilities to each state–action pair.

For any $\pi \in \Pi_R$, we can define the transition probability matrix and the reward vector as follows: $P_\pi(s,s') = \sum_{a\in\mathcal{A}} P(s,a,s')\cdot\pi(s,a)$ and $r_\pi(s) = \sum_{a\in\mathcal{A}} r(s,a)\cdot\pi(s,a)$. We use P_a and r_a to represent values for a policy that always takes action $a \in \mathcal{A}$. We also define

a matrix A and a vector b as follows:

$$A^\mathsf{T} = \begin{pmatrix} \mathbf{I} - \gamma P_{a_1}^\mathsf{T} & \mathbf{I} - \gamma P_{a_2}^\mathsf{T} & \cdots \end{pmatrix}, \quad b^\mathsf{T} = \begin{pmatrix} r_{a_1}^\mathsf{T} & r_{a_2}^\mathsf{T} & \cdots \end{pmatrix}$$

Values A and b are usually used in approximate linear programming (ALP) (Schweitzer & Seidmann, 1985) and linear program formulations of MDPs (Puterman, 2005). The main objective in solving an MDP is to compute a policy with the maximal return.

Definition 2.1. The return $\rho : \Pi_R \to \mathbb{R}$ of $\pi \in \Pi_R$ is defined as: $\rho(\pi) = \sum_{n=0}^{\infty} \alpha^\mathsf{T} (\gamma \cdot P_\pi)^n \, r_\pi$. The *optimal policy* solves $\pi^\star \in \arg\max_{\pi \in \Pi_R} \rho(\pi)$ and we use $\rho^\star = \rho(\pi^\star)$.

DRADP relies on two main solution concepts: *state occupancy frequencies* and *value functions*. State occupancy frequencies—or measures—intuitively represent the probability of terminating in each state when the discount factor γ is interpreted as a probability of remaining in the system (Puterman, 2005). State-action occupancy frequencies are defined for state–action pairs and represent the joint probability of being in the state and taking the action.

State occupancy frequency for $\pi \in \Pi_R$ is denoted by $d_\pi \in \mathbb{R}^\mathcal{S}$ and is defined as:

$$d_\pi = (1-\gamma) \cdot \sum_{t=0}^{\infty} (\gamma \cdot P_\pi^\mathsf{T})^t \alpha = (1-\gamma) \cdot \left(\mathbf{I} - \gamma \cdot P_\pi^\mathsf{T}\right)^{-1} \alpha \, .$$

State-action occupancy frequency is denoted by $u_\pi \in \mathbb{R}^\mathcal{W}$(its set-valued equivalent is $U(\pi)$) and is a product of state–occupancy frequencies and action probabilities:

$$u_\pi(s,a) = d_\pi(s) \cdot \pi(s,a) \, , \quad U(\pi) = \{u_\pi\} \, .$$

Note that $U(\pi)$ is a set-valued function with the output set of cardinality 1. State and state-action occupancy frequencies represent valid probability measures over $\mathcal{S}$ and $\mathcal{W}$ respectively. We use $d^\star = d_{\pi^\star}$ and $u^\star = u_{\pi^\star}$ to denote the optimal measures. Finally, we use $u|_\pi \in \mathbb{R}_+^\mathcal{S}$ for $\pi \in \Pi_D$ to denote a restriction of u to π such that $u|_\pi(s) = u(s, \pi(s))$.

State-action occupancy frequencies are closely related to the set $\mathcal{U}$ of dual feasible solutions of the linear program formulation of an MDP, which is defined as (e.g. Section 6.9 of (Puterman, 2005)):

$$\mathcal{U} = \left\{ u \in \mathbb{R}_+^\mathcal{W} : A^\mathsf{T} u = (1-\gamma) \cdot \alpha \right\}. \tag{2.1}$$

The following well-known proposition characterizes the basic properties of the set $\mathcal{U}$.

Proposition 2.2 (e.g. Theorem 6.9 in (Puterman, 2005))**.** *The set of occupancy frequencies satisfies the following properties.*

(i) $\mathcal{U} = \bigcup_{\pi \in \Pi_R} U(\pi) = \mathrm{conv}(\bigcup_{\pi \in \Pi_D} U(\pi))$.
(ii) For each $\bar{u} \in \mathcal{U}$, *define* $\pi'(s,a) = \bar{u}(s,a) / \sum_{a' \in \mathcal{A}} \bar{u}(s,a')$. *Then* $u_{\pi'} = \bar{u}$.
(iii) $\mathbf{1}^\mathsf{T} u = 1$ *for each* $u \in \mathcal{U}$.
(iv) $A^\mathsf{T} u = (1-\gamma) \cdot \alpha$ *for each* $u \in \mathcal{U}$.

Part (i), in particular, holds because deterministic policies represent the basic feasible solutions of the dual linear program for an MDP.

A *value function* $v_\pi \in \mathbb{R}^\mathcal{S}$ of $\pi \in \Pi_R$ maps states to the return obtained when starting in them and is defined by:

$$v_\pi = \sum_{t=0}^{\infty} (\gamma \cdot P_\pi)^t r_\pi = \left(\mathbf{I} - \gamma \cdot P_\pi\right)^{-1} r_\pi \, .$$

The set of all possible value functions is denoted by $\mathcal{V}$. It is well known that a policy $\pi^\star$ with the value function $v^\star$ is *optimal* if and only if $v^\star \geq v_\pi$ for every $\pi \in \Pi_R$. The value function update L_π for a policy π and the Bellman operator L are defined as: $L_\pi v = \gamma P_\pi v + r_\pi$ and $Lv = \max_{\pi \in \Pi_R} L_\pi v$.

The optimal value function $v^\star$ satisfies $Lv^\star = v^\star$. The following proposition states the well-known connection between state–action occupancy frequencies and value functions.

Proposition 2.3 (e.g. Chapter 6 in (Puterman, 2005))**.** *For each* $\pi \in \Pi_R$*:* $\rho(\pi) = \mathbf{E}_\alpha \left[v_\pi(S)\right] = \mathbf{E}_{u_\pi} \left[r(W)\right] / (1-\gamma)$.

The value function, computed by a dynamic programming algorithm, is typically then used to derive the greedy policy. A greedy policy takes in every state an action that maximizes the expected conditional return.

Definition 2.4. A policy $\pi \in \Pi_D$ is *greedy* with respect to a value function v when $L_\pi v = Lv$; in other words:

$$\pi(s) \in \arg\max_{a \in \mathcal{A}} \Big(\, r(s,a) + \gamma \cdot \sum_{s' \in \mathcal{S}} P(s,a,s') \cdot v(s') \, \Big),$$

for each $s \in \mathcal{S}$ with ties broken arbitrarily.

MDP is a very general model. Often, specific properties of the MDP can be used to compute better solutions and to derive tighter bounds. One common assumption—used to derive L_2 bounds for API—is a smoothness of transition probabilities (Munos, 2003), also known as the concentration coefficient (Munos, 2007); this property can be used to derive tighter DRADP bounds.

Assumption 1 (Concentration coefficient)**.** There exists a probability measure $\mu \in [0,1]^\mathcal{S}$ and a constant $C \in \mathbb{R}_+$ such that for all $s, s' \in \mathcal{S}$ and all

$\pi \in \Pi_D$ the transition probability is bounded as: $P(s, \pi(s), s') \leq C \cdot \mu(s')$.

3. Distributionally Robust Approximate Dynamic Programming

In this section, we formalize DRADP and describe it in terms of generic optimization problems. Practical DRADP implementations are sampled versions of the optimization problems described in this section. However, as it is common in ADP literature, we do not explicitly analyze the sampling method used with DRADP in this paper, because the sampling error can simply be added to the error bounds that we derive. The sampling is performed and errors bounded identically to approximate linear programming and approximate bilinear programming—state and action samples are used to select a subset of constraints and variables (de Farias & van Roy, 2003; Petrik et al., 2010; Petrik & Zilberstein, 2011).

The main objective of ADP is to compute a policy $\pi \in \Pi_R$ that maximizes the return $\rho(\pi)$. Because the MDPs of interest are very large, a common approach is to simplify them by restricting the set of policies that are considered to a smaller set $\tilde{\Pi} \subseteq \Pi_R$. For example, policies may be constrained to take the same action in some states; or to be greedy with respect to an approximate value function. Since it is not possible to compute an optimal policy, the common objective is to minimize the policy loss. Policy loss captures the difference in the discounted return when following policy π instead of the optimal policy $\pi^\star$.

Definition 3.1. The *expected policy loss* of $\pi \in \Pi_R$ is defined as:

$$\rho^\star - \rho(\pi) = \frac{b^\mathsf{T}(u^\star - u_\pi)}{1-\gamma} = \|v^\star - v_\pi\|_{1,\alpha} \ ,$$

where $\|\cdot\|_{1,\alpha}$ represents an α-weighted L_1 norm.

ADP relies on approximate value functions $\tilde{\mathcal{V}} \subseteq \mathcal{V}$ that are a *subset* of all value functions. In DRADP, approximate value functions are used simultaneously to both restrict the space of policies and to approximate their returns. We, in addition, define a set of approximate occupancy frequencies $\tilde{U}(\pi) \supseteq U(\pi)$ that are a *superset* of the true occupancy frequencies. We call any element in the appropriate approximate sets *representable*.

We consider *linear function approximation*, in which the values for states are represented as a linear combination of *nonlinear basis functions (vectors)*. For each $s \in \mathcal{S}$, we define a vector $\phi(s)$ of features with $|\phi|$ being the dimension of the vector. The rows of the basis matrix Φ correspond to $\phi(s)$, and the approximation space is generated by the columns of Φ. Approximate value functions and *policy-dependent* state occupancy measures for linear approximations are defined for some given feature matrices Φ_u and Φ_v as:

$$\tilde{\mathcal{V}} = \left\{ \ v \in \mathcal{V} : v = \Phi_v x, \ x \in \mathbb{R}^{|\phi|} \ \right\}, \tag{3.1}$$

$$\tilde{U}(\pi) = \left\{ \ u \in \mathbb{R}_+^{\mathcal{A}} : \begin{matrix} \Phi_u^\mathsf{T} A^\mathsf{T} u = (1-\gamma) \cdot \Phi_u^\mathsf{T} \alpha, \\ u(s,a) \leq \pi(s,a) \end{matrix} \right\}. \tag{3.2}$$

Clearly, $\tilde{U}(\pi) \supseteq U(\pi)$ from the definition of u_π. We will assume the following important assumption without reference for the remainder of the paper.

Assumption 2. One of the features in each of ϕ_u and ϕ_v is a constant; that is, $\mathbf{1} = \Phi_u x_u$ and $\mathbf{1} = \Phi_v x_v$ for some x_u and x_v.

The following lemma, which can be derived directly from the definition of $\tilde{U}$ and Proposition 2.2, shows the importance of Assumption 2.

Lemma 3.2. *Suppose that Assumption 2 holds. Then for each $\pi \in \Pi_R$: $u \in \tilde{U}(\pi) \Rightarrow \mathbf{1}^\mathsf{T} u = 1$.*

Approximate policies $\tilde{\Pi}$ are most often represented indirectly—by assuming policies that are greedy to the approximate value functions. The set $\mathcal{G}$ of all such greedy policies is defined by: $\mathcal{G} = \{ \ \pi \in \Pi_D : L_\pi v = Lv, v \in \tilde{\mathcal{V}} \ \}$. Although DRADP applies to other approximate policy sets we will particularly focus on the set $\tilde{\Pi} = \mathcal{G}$.

We are now ready to define the basic DRADP formulation which is analyzed in the remainder of the paper.

Definition 3.3. DRADP computes an approximate policy by solving the following optimization problem:

$$\arg\max_{\pi \in \tilde{\Pi}} \tilde{\rho}(\pi) = \arg\min_{\pi \in \tilde{\Pi}} \left(\ \rho^\star - \tilde{\rho}(\pi) \ \right), \tag{DRADP}$$

where the function $\tilde{\rho} : \Pi_R \to \mathbb{R}$ is defined by:

$$\tilde{\rho}(\pi) = \max_{v \in \tilde{\mathcal{V}}} \left(\ \alpha^\mathsf{T} v - \max_{u \in \tilde{U}(\pi)} \frac{u^\mathsf{T}(Av - b)}{1-\gamma} \ \right). \tag{3.3}$$

Note that the solution of (DRADP) is a policy; this policy is not necessarily greedy with respect to the optimal v in (3.3) unlike in most other ADP approaches. The expression (3.3) can be understood intuitively as follows. The first term, $\alpha^\mathsf{T} v$, represents the expected return if v is the value function of π. The second term $\max_{u \in \tilde{U}(\pi)}(u^\mathsf{T}(Av - b))/(1-\gamma)$ is a penalty function, which offsets any gains when $v \neq v_\pi$ and is motivated by the primal-dual slack variables in the LP formulation of the MDP. Given this interpretation, DRADP

simultaneously restricts the set of value functions and upper-approximates the penalty function.

The following theorem states an important property of Definition 3.3, which is used to derive approximation error bounds.

Theorem 3.4. *For each $\pi \in \Pi_R$, $\tilde{\rho}$ lower-bounds the true return: $\tilde{\rho}(\pi) \leq \rho(\pi)$. In addition, when Φ_u and Φ_v are invertible and $\pi \in \Pi_D$ then $\rho(\pi) = \tilde{\rho}(\pi)$.*

We now show that the lower bound $\tilde{\rho}$ in (3.3) can be simplified in some cases by ignoring the value functions for any $\pi \in \Pi_R$; the formulation (3.3) will nevertheless be particularly useful in the theoretical analysis because it relates value functions and occupancy frequencies.

$$\tilde{\rho}'(\pi) = \min_{u \in \tilde{U}'(\pi)} \frac{u^\mathsf{T} b}{1-\gamma} \ , \tag{3.4}$$

where $\tilde{U}'(\pi)$ is defined equivalently to $\tilde{U}(\pi)$ with the exception that $\Phi_u = \Phi_v = \Phi$ for some Φ.

Proposition 3.5. *When $\Phi_v = \Phi_u$, then $\tilde{\rho}(\pi) = \tilde{\rho}'(\pi)$. When $\Phi_v \neq \Phi_u$, then define $\tilde{U}'$ and $\tilde{\rho}'$ using a new representation $\Phi' = [\Phi_v \ \Phi_u]$. Then: $\tilde{\rho}(\pi) = \tilde{\rho}'(\pi)$.*

For the remainder of the paper assume that $\Phi_v = \Phi_u$ since assuming that they are the same does not reduce the solution quality.

A potential challenge with DRADP is in representing the set of approximate policies $\tilde{\Pi}$, because a policy must generalize to all states even when computed from a small sample. Note, that for a fixed value function v in (3.3) the policy that solves $\min_{\pi \in \tilde{\Pi}} \tilde{\rho}(\pi)$ is not necessarily the greedy with respect to v. The following representation theorem, however, shows that when the set of representable policies $\tilde{\Pi}$ is sufficiently rich, then the computed policy will be greedy with respect to a representable value function.

Theorem 3.6. *Suppose that $\tilde{\Pi} \supseteq \mathcal{G}$. Then:*

(i) $\max_{\pi \in \tilde{\Pi}} \tilde{\rho}(\pi) = \max_{\pi \in \mathcal{G}} \tilde{\rho}(\pi)$.
(ii) $\exists \bar{\pi} \in \arg\max_{\pi \in \tilde{\Pi}} \tilde{\rho}(\pi)$ *such that* $\bar{\pi} \in \mathcal{G}$.

Note that the assumption $\tilde{\Pi} \supseteq \mathcal{G}$ simply implies that DRADP can select a policy that is greedy with respect to any *approximate* value function $v \in \tilde{\mathcal{V}}$. This is an implicit assumption in many ADP algorithms, including ALP and LSPI. We state the assumption explicitly to indicate results that do not hold in case there are additional restrictions on the set of policies that is considered.

Theorem 3.6 implies that it is only necessary to consider policies that are greedy with respect to representable value functions which is the most common approach in ADP. However, other approaches for representing policies may have better theoretical or empirical properties and should be also studied.

4. Approximation Error Bounds

This section describes the *a priori* approximation properties of DRADP solutions; these bounds can be evaluated before a solution is computed. We focus on several types of bounds that not only show the performance of the method, but also make it easier to theoretically compare DRADP to existing ADP methods. These bounds show that DRADP has stronger theoretical guarantees than most other ADP methods. The first bound mirrors some simple bounds for approximate policy iteration (API) in terms of the L_∞ norm (Munos, 2007):

$$\limsup_{k \to \infty} \|v^\star - v_{\pi_k}\|_\infty \leq \frac{2 \cdot \gamma}{(1-\gamma)^2} \limsup_{k \to \infty} \epsilon_k \ , \tag{4.1}$$

where π_k and ϵ_k are the policy and L_∞ approximation error at iteration k.

Theorem 4.1. *Suppose that $\tilde{\Pi} \supseteq \mathcal{G}$ and that $\bar{\pi} \in \arg\max_{\pi \in \tilde{\Pi}} \tilde{\rho}(\pi)$ in (DRADP). The policy loss $\rho^\star - \rho(\bar{\pi})$ is then bounded as:*

$$\|v^\star - v_{\bar{\pi}}\|_{1,\alpha} \leq \frac{2}{1-\gamma} \min_{v \in \tilde{\mathcal{V}}} \|v - Lv\|_\infty \ . \tag{4.2}$$

Theorem 4.1 highlights several advantages of the DRADP bound (4.2) over (4.1): 1) it bounds the expected loss $\|v^\star - v_{\bar{\pi}}\|_{1,\alpha}$ instead of the worst-case loss $\|v^\star - v_{\bar{\pi}}\|_\infty$, 2) it is smaller by a factor of $1/(1-\gamma)$, 3) it holds in finite time instead of a limit, and 4) its right-hand side is with respect to the best approximation of the optimal value function instead of the worst case approximation over all iteration. In comparison with approximate linear programming bounds, (4.2) bounds the true policy loss and not simply the approximation of $v^\star$ (de Farias & van Roy, 2003). The limitation of (4.2), however, is that it relies on an L_∞ norm which can be quite conservative. We address this issue in two ways. First, we prove a bound of a different type.

Theorem 4.2. *Suppose that $\tilde{\Pi} \supseteq \mathcal{G}$ and that $\bar{\pi} \in \arg\max_{\pi \in \tilde{\Pi}} \tilde{\rho}(\pi)$ in (DRADP). Then, the policy loss $\rho^\star - \rho(\bar{\pi})$ is bounded as:*

$$\|v^\star - v_{\bar{\pi}}\|_{1,\alpha} \leq \min_{v \in \tilde{\mathcal{V}}, v \leq v^\star} \|v - v^\star\|_{1,\alpha} \ . \tag{4.3}$$

The bound (4.3), unlike bounds in most ADP algorithms, does not contain a factor of $1/(1-\gamma)$ of any power. Although (4.3) does not involve an L_∞ norm,

it does require that $v \leq v^\star$ which may be undesirable. Next, we show bounds that rely purely on weighted norms under additional assumptions on the concentration coefficient.

As mentioned above, Assumption 1 can be used to improve the solutions of DRADP and to derive tighter bounds. Note that this assumption must be known in advance and cannot be gleaned from the samples. To this end, for some fixed $C \in \mathbb{R}_+$ and $\mu \in \mathbb{R}^{\mathcal{S}}$ in Assumption 1, define:

$$\tilde{U}_S(\pi) = \left\{ u \in \tilde{U}(\pi) : \sum_{a \in \mathcal{A}} u(s,a) \leq \sigma(s),\ \forall s \in \mathcal{S} \right\},$$
$$\sigma(s) = \gamma \cdot \mu(s) + (1-\gamma) \cdot \alpha(s),$$
$$\tilde{\rho}_S(\pi) = \max_{v \in \tilde{\mathcal{V}}} \min_{u \in \tilde{U}_S(\pi)} \left(\alpha^\mathsf{T} v - \frac{u^\mathsf{T}(Av - b)}{1-\gamma} \right).$$

These assumptions imply the following structure of all admissible state frequencies.

Lemma 4.3. *Suppose that Assumption 1 holds with constants C and μ. Then: $d \leq C \cdot \sigma$ for each $d \in \tilde{U}_S(\pi)$ and $\pi \in \Pi_R$.*

The following theorem shows a tighter bound on the DRADP policy loss for MDPs that satisfy the smoothness assumption.

Theorem 4.4. *Suppose that Assumption 1 holds with constants C and μ, $\tilde{\Pi} \supseteq \mathcal{G}$, and that $\bar{\pi} \in \arg\max_{\pi \in \tilde{\Pi}} \tilde{\rho}(\pi)$ in (DRADP). Then, the loss of $\bar{\pi}$ is bounded as:*

$$\rho^\star - \rho(\bar{\pi}) \leq \frac{2 \cdot C}{1-\gamma} \min_{v \in \tilde{\mathcal{V}}} \|v - Lv\|_{1,\sigma} \ . \tag{4.4}$$

The bound in Theorem 4.4 is similar to comparable L_p bounds for API (Munos, 2003), except it relies on a weighted L_1 norm instead of the L_2 norm and preserves all the advantages of Theorem 4.1. Theorem 4.4 exploits that the set of occupancy frequencies is restricted under the smoothness assumption which leads to a tighter lower bound $\tilde{\rho}_S$ on the return.

Finally, DRADP is closely related to robust ABP (Petrik & Zilberstein, 2009; 2011) but provides several significant advantages. First, DRADP does not require transitive feasible (Petrik & Zilberstein, 2011) value functions, which simplifies the use of constraint generation. Second, ABP minimizes L_∞ bounds $\rho_r : \Pi_R \to \mathbb{R}$ on the policy loss, which can be too conservative. In fact, it is easy to show that DRADP solutions can be better than ABP solutions by an arbitrarily large factor.

5. Computational Models

In this section, we describe how to solve the DRADP optimization problem. Since DRADP generalizes ABP (Petrik & Zilberstein, 2009), it is necessarily NP complete to solve in theory, but relatively easy to solve in practice. Note that the NP-completeness is in terms of the number of samples and features and not in the number of states or actions of the MDP. In addition, the NP completeness a is favorable property when compared to API algorithms, such as LSPI, which may never converge (Lagoudakis & Parr, 2003).

To solve DRADPs in practice, we derive bilinear and mixed integer linear program formulations for which many powerful solvers have been developed. These formulations lead to anytime solvers—even approximate solutions result in valid policies—and can therefore easily trade off solution quality with time complexity.

To derive bilinear formulations of DRADP, we represent the set of policies $\tilde{\Pi}$ using linear equalities as: $\tilde{\Pi} = \left\{ \pi \in [0,1]^{\mathcal{W}} : \sum_{a \in \mathcal{A}} \pi(s,a) = 1 \right\}$. This set can be defined using matrix notation as $B\pi = \mathbf{1}$ and $\pi \geq \mathbf{0}$, where $B : |\mathcal{S}| \times |\mathcal{W}|$ is defined as: $B(s',(s',a)) = 1$ when $s = s'$ and 0 otherwise. Clearly $\tilde{\Pi} \supseteq \mathcal{G}$, which implies that the computed policy is greedy with respect to a representable value function from Theorem 3.6 even as sampled. It would be easy to restrict the set $\tilde{\Pi}$ by assuming the same action must be taken in a subset of states: one would add constraints $\pi(s,a) = \pi(s',a)$ for some $s, s' \in \mathcal{S}$ and all $a \in \mathcal{A}$.

When the set of approximate policies is represented by linear inequalities, the DRADP optimization problem can be formulated as the following *separable bilinear program* (Horst & Tuy, 1996).

$$\begin{array}{lll} \max\limits_{\pi, \lambda_1, \lambda_2} & \alpha^\mathsf{T} \Phi \lambda_1 - \pi^\mathsf{T} \lambda_2 & \\ \text{s.t.} & B\pi = \mathbf{1}, \quad \pi \geq \mathbf{0}, \quad \lambda_2 \geq \mathbf{0}, & (5.1) \\ & (1-\gamma) \cdot \lambda_2 \geq A\Phi\lambda_1 - b \ . & \end{array}$$

Bilinear programs are a generalization of linear programs and are in general NP hard to solve.

Theorem 5.1. *Suppose that $\tilde{\Pi} \supseteq \mathcal{G}$. Then the sets of optimal solutions of (5.1) and (DRADP) are identical and there exists an optimal solution $(\bar{\pi}, \bar{\lambda}_1, \bar{\lambda}_2)$ of (5.1) such that:*

(i) $\bar{\pi}$ is deterministic and greedy with respect to $\Phi\bar{\lambda}_1$,
(ii) $\bar{\pi}^\mathsf{T}\bar{\lambda}_2 = 0$.

Because there are few, if any, industrial solvers for bilinear programs, we reformulate (5.1) as a mixed integer linear program (MILP). Any separable bilinear program can be formulated as a MILP (Horst &

Tuy, 1996), but such generic formulations are impractical because they lead to large MILPs with weak linear relaxations. Instead, we derive a more compact and structured MILP formulation that exploits the existence of optimal deterministic policies in DRADP (see (i) of Theorem 5.1) and is based on McCormic inequalities on the bilinear terms (Linderoth, 2005). To formulate the MILP, assume a given upper bound $\tau \in \mathbb{R}$ on any optimal solution $\lambda_2{}^\star$ of (5.1) such that $\tau \geq \lambda_2{}^\star(s,a)$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}$. Then:

$$\begin{aligned} \max_{z,\pi,\lambda_1,\lambda_2} \quad & \alpha^\mathsf{T}\Phi\lambda_1 - \mathbf{1}^\mathsf{T} z \\ \text{s.t.} \quad & z \geq \lambda_2 - \tau \cdot (\mathbf{1} - \pi), \\ & (1-\gamma)\cdot\lambda_2 \geq A\Phi\lambda_1 - b, \\ & B\pi = \mathbf{1}, \quad \pi \in \{0,1\}^{|\mathcal{S}||\mathcal{A}|} \\ & z \geq \mathbf{0}, \quad \lambda_2 \geq \mathbf{0}\ . \end{aligned} \tag{5.2}$$

Theorem 5.2. *Suppose that* $\tilde{\Pi} \supseteq \mathcal{G}$ *and* $(\bar{\pi}, \bar{\lambda}_1, \bar{\lambda}_2, \bar{z})$ *is an optimal solution of* (5.2)*. Then,* $(\bar{\pi}, \bar{\lambda}_1, \bar{\lambda}_2)$ *is an optimal solution of* (5.1) *with the same objective value given that* $\tau > \|\bar{\lambda}_2\|_\infty$.

As discussed above, any practical implementation of DRADP must be sample-based. The bilinear program (5.1) is constructed from samples very similarly to ALPs (e.g. Sec 6 of (de Farias & van Roy, 2003)) and identically to ABPs (e.g. Sec 6 of (Petrik & Zilberstein, 2011)). Briefly, the formulation involves only the rows of A that correspond to transitions of sampled state-action pairs and b entries are estimated from the corresponding rewards. As a result, there is one λ_1 variable for each feature, and λ_2 and π are nonzero only for the sampled rows of A (zeros do not need to be considered). The size of the optimization problem (5.1) is then independent of the number of states and actions of the MDP; it depends only on the number of samples and features.

6. Experimental Results

In this section, we experimentally evaluate the empirical performance of DRADP. We present results on the inverted pendulum problem—a standard benchmark problem—and a synthetic chain problem. We gather state and action samples in advance and solve MILP (5.2) using IBM CPLEX 12.2. We then compare the results to three related methods which work on offline samples: 1) LSPI (Lagoudakis & Parr, 2003), 2) ALP (de Farias & van Roy, 2003), and 3) ABP (Petrik & Zilberstein, 2009). While solving the MILP formulation of DRADP is NP hard (in the number of features and samples), this does not mean that the computation takes longer than for other ADP methods; for example, the computation time of LSPI is unbounded in the worst case (there are no convergence guarantees). In the experiments, we restrict the computation time for all methods to 60s.

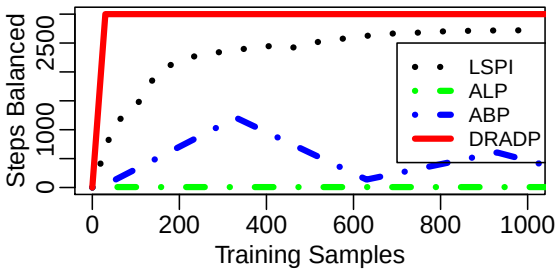

Figure 1. Inverted pendulum results

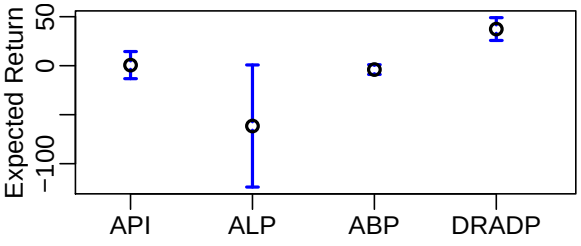

Figure 2. Expected return on the chain benchmark

Inverted Pendulum The goal in the inverted pendulum benchmark problem is to balance an inverted pole by accelerating a cart in either of two directions (Lagoudakis & Parr, 2003). There are three actions that represent applying the force of $u = -50N$, $u = 0N$, and $u = 50N$ to the cart with a uniform noise between $-10N$ and $10N$. The angle of the inverted pendulum is governed by a differential equation. We used the standard features for this benchmark problem for all the methods: 9 radial basis functions arranged in a grid over the 2-dimensional state space with centers μ_i and a constant term required by Assumption 2. The problem setting, including the initial distribution is identical to the setting in (Lagoudakis & Parr, 2003).

Fig. 1 shows the number of balancing steps (with a 3000-step bound) for each method as a function of the number of training samples averaged over 5 runs. The figure does not show error bars for clarity; the variance was close to 0 for DRADP. The results indicate that DRADP computes a very good solution for even a small number of samples and significantly outperforms LSPI. Note the poor performance of ABP and ALP with the 10 standard features; better results have been obtained with large and different feature spaces (Petrik

et al., 2010) but even these do not match DRADP. The solution quality of ABP decreases with more samples, because the bounds become more conservative and the optimization problems become harder to solve.

Chain Problem Because the solution quality of ADP methods depends on many factors, good results on a single benchmark problem do not necessarily generalize to other domains. We, therefore, compare DRADP to other methods on a large number of randomly generated chain problems. This problem consists of 30 states $s_1 \dots s_{30}$ and 2 actions: left and right with 10% chance of moving the opposite way. The features are 10 orthogonal polynomials. The rewards are 0 except: $r(s_2) = -50,\ r(s_3) = 4,\ r(s_4) = -50,\ r(s_{20}) = 10$. Fig. 2 shows the results of 1000 instantiations with randomly chosen initial distributions and indicates that DRADP significantly outperforms other methods including API (a simple version of LSPI).

7. Conclusion

This paper proposes and analyzes DRADP—a new ADP method. DRADP is based on a mathematical optimization formulation—like ALP—but offers significantly stronger theoretical guarantees and better empirical performance. The DRADP framework also makes it easy to improve the solution quality by incorporating additional assumptions on state occupation frequencies, such as the small concentration coefficient. Given the encouraging theoretical and empirical properties of DRADP, we hope it will lead to better methods for solving large MDPs and will help to deepen the understanding of ADP.

Acknowledgements I thank Dan Iancu and Dharmashankar Subramanian for the discussions that inspired this paper. I also thank the anonymous ICML 2012 and EWRL 2012 reviewers for their detailed comments.

References

Ben-Tal, Aharon, Ghaoui, Laurent El, and Nemirovski, Arkadi. *Robust Optimization*. Princeton University Press, 2009.

de Farias, Daniela P. and van Roy, Ben. The linear programming approach to approximate dynamic programming. *Operations Research*, 51:850–856, 2003.

Delage, Eric and Ye, Yinyu. Distributionally robust optimization under moment uncertainty with application to data driven problems. *Operations Research*, 58(3): 592–612, 2010.

Dolgov, Dmitri and Durfee, Edmund. Symmetric approximate linear programming for factored MDPs with application to constrained problems. *Annals of Mathematics and Artificial Intelligence*, 47(3-4):273–293, 2006.

Horst, Reiner and Tuy, Hoang. *Global optimization: Deterministic approaches*. Springer, 1996.

Lagoudakis, Michail G. and Parr, Ronald. Least-squares policy iteration. *Journal of Machine Learning Research*, 4:1107–1149, 2003.

Linderoth, Jeff. A simplicial branch-and-bound algorithm for solving quadratically constrained quadratic programs. *Mathematical Programming, Series B*, 103: 251–282, 2005.

Munos, Remi. Error bounds for approximate policy iteration. In *International Conference on Machine Learning*, pp. 560–567, 2003.

Munos, Remi. Performance bounds in Lp norm for approximate value iteration. *SIAM Journal of Control and Optimization*, 46:541–561, 2007.

Petrik, Marek. Approximate dynamic programming by minimizing distributionally robust bounds. Arxiv:1205.1782, 2012.

Petrik, Marek and Zilberstein, Shlomo. Robust value function approximation using bilinear programming. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 1446–1454, 2009.

Petrik, Marek and Zilberstein, Shlomo. Robust approximate bilinear programming for value function approximation. *Journal of Machine Learning Research*, 12: 3027–3063, 2011.

Petrik, Marek, Taylor, Gavin, Parr, Ron, and Zilberstein, Shlomo. Feature selection using regularization in approximate linear programs for Markov decision processes. In *International Conference on Machine Learning*, pp. 871–878, 2010.

Puterman, Martin L. *Markov decision processes: Discrete stochastic dynamic programming*. John Wiley & Sons, Inc., 2005.

Schweitzer, Paul J. and Seidmann, Abraham. Generalized polynomial approximations in Markovian decision processes. *Journal of Mathematical Analysis and Applications*, 110:568–582, 1985.

Wang, Tao. *New Representations and Approximations for Sequential Decision Making*. PhD thesis, University of Alberta, Canada, 2007.

Wang, Tao, Lizotte, Daniel, Bowling, Michael, and Schuurmans, Dale. Stable dynamic programming. In *Neural Information Processing Systems (NIPS)*, pp. 1569–1576, 2008.