Journal of Machine Learning Research 12 (2011) 3027-3063 Submitted 4/10; Revised 1/11; Published 10/11

Robust Approximate Bilinear Programming for Value Function Approximation

Marek Petrik MPETRIK@US.IBM.COM
IBM T.J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598, USA

Shlomo Zilberstein SHLOMO@CS.UMASS.EDU
Department of Computer Science
University of Massachusetts
Amherst, MA 01003, USA

Editor: Shie Mannor

Abstract

Value function approximation methods have been successfully used in many applications, but the prevailing techniques often lack useful *a priori* error bounds. We propose a new *approximate bilinear programming* formulation of value function approximation, which employs global optimization. The formulation provides strong a priori guarantees on both robust and expected policy loss by minimizing specific norms of the Bellman residual. Solving a bilinear program optimally is NP-hard, but this worst-case complexity is unavoidable because the Bellman-residual minimization itself is NP-hard. We describe and analyze the formulation as well as a simple approximate algorithm for solving bilinear programs. The analysis shows that this algorithm offers a *convergent* generalization of approximate policy iteration. We also briefly analyze the behavior of bilinear programming algorithms under incomplete samples. Finally, we demonstrate that the proposed approach can consistently minimize the Bellman residual on simple benchmark problems.

Keywords: value function approximation, approximate dynamic programming, Markov decision processes

1. Introduction

Solving large Markov Decision Processes (MDPs) is a very useful, but computationally challenging problem addressed widely in the AI literature, particularly in the area of reinforcement learning. It is widely accepted that large MDPs can be solved only approximately. The commonly used approximation methods can be divided into three broad categories: 1) *policy search*, which explores a restricted space of all policies, 2) *approximate dynamic programming*—or value function approximation—which searches a restricted space of value functions, and 3) *approximate linear programming*, which approximates the solution using a linear program. The goal of approximate methods is to compute a policy that minimizes the *policy loss*—the difference between the returns of the computed policy and an optimal one. While all of these approximate methods have achieved impressive results in various application domains, they have significant limitations.

Policy search methods rely on local search in a restricted policy space. The policy may be represented, for example, as a finite-state controller (Stanley and Miikkulainen, 2004) or as a greedy policy with respect to an approximate value function (Szita and Lorincz, 2006). Policy search

methods have achieved impressive results in such domains as Tetris (Szita and Lorincz, 2006) and helicopter control (Abbeel et al., 2006). However, they are notoriously hard to analyze. We are not aware of any established theoretical guarantees regarding the quality of the solution.

Approximate dynamic programming (ADP) methods iteratively approximate the value function (Bertsekas and Ioffe, 1997; Powell, 2007; Sutton and Barto, 1998). They have been extensively analyzed and are the most commonly used methods. However, approximate dynamic programming methods typically do not converge and they only provide weak guarantees of approximation quality. The approximation error bounds are usually expressed in terms of the worst-case approximation of the value function over all policies (Bertsekas and Ioffe, 1997). In addition, most available bounds are with respect to the L_∞ norm, while the algorithms often minimize the L_2 norm. While there exist some L_2-based bounds (Munos, 2003), they require values that are difficult to obtain.

Approximate linear programming (ALP) uses a linear program to compute the approximate value function in a particular vector space (de Farias, 2002). ALP has been previously used in a wide variety of settings (Adelman, 2004; de Farias and van Roy, 2004; Guestrin et al., 2003). Although ALP often does not perform as well as ADP, there have been some recent efforts to close the gap (Petrik and Zilberstein, 2009). ALP has better theoretical properties than ADP and policy search. It is guaranteed to converge and return the closest L_1-norm approximation $\tilde{v}$ of the optimal value function $v^\star$ up to a multiplicative factor. However, the L_1 norm must be properly weighted to guarantee a small policy loss, and there is no *reliable* method for selecting appropriate weights (de Farias, 2002).

To summarize, existing reinforcement learning techniques often provide good solutions, but typically require significant domain knowledge (Powell, 2007). The domain knowledge is needed partly because useful a priori error bounds are not available, as mentioned above. Our goal is to develop a more *reliable* method that is guaranteed to minimize bounds on the policy loss in various settings.

This paper presents new formulations for value function approximation that provably minimize bounds on policy loss using a global optimization framework; we consider both L_∞ and weighted L_1 error bounds. To minimize the policy loss, we derive new bounds based on approximate value functions. These bounds do not require coefficients that are hard to obtain or compute, unlike, for example, bounds for approximate linear programming.

An advantage of the approach we propose is that the actual solutions and their properties are independent of the methods used to compute them. The paper focuses on the development of models for value function approximation and their properties. Although we do present two methods for solving these models, it is likely that more efficient algorithms will be developed in the future.

We start with a description of the framework and notation in Section 2 and a description of value function approximation in Section 3. Then, in Section 4, we describe the proposed approximate bilinear programming (ABP) formulations. Bilinear programs are typically solved using global optimization methods, which we briefly discuss in Section 5. A drawback of the bilinear formulation is that solving bilinear programs may require exponential time. We show in Section 5, however, that this complexity is unavoidable because minimizing the approximation error bound is in fact NP-hard.

In practice, only sampled versions of ABPs are often solved. While a thorough treatment of sampling is beyond the scope of this paper, we examine the impact of sampling and establish some guarantees in Section 6. Unlike classical sampling bounds on approximate linear programming, we describe bounds that apply to the worst-case error. Section 7 shows that ABP is related to other

approximate dynamic programming methods, such as approximate linear programming and policy iteration. Section 8 demonstrates the applicability of ABP using common reinforcement learning benchmark problems.

The general setting considered in this paper is a restricted form of reinforcement learning. In reinforcement learning, methods can use samples without requiring a model of the environment. The methods we propose can also be based on samples, but they require additional structure. In particular, they require that all or most actions are sampled for every state. Such samples can be easily generated when a model of the environment is available.

2. Framework and Notation

This section defines the framework and the notation we use. We also define Markov decision processes and the associated approximation errors. Markov decision processes come in many forms, depending on the objective function that is optimized. This work focuses on infinite-horizon discounted MDPs, which are defined as follows; a more extensive treatment is available, for example, in Puterman (2005).

Definition 1 *A* Markov Decision Process *is a tuple* $(\mathcal{S}, \mathcal{A}, P, r, \alpha)$*, where* $\mathcal{S}$ *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 goal in solving an MDP is to find a sequence of actions that maximizes the expected γ-discounted cumulative sum of rewards, also called the *return*. A solution of a Markov decision process is a policy, defined as follows.

Definition 2 *A deterministic stationary* policy $\pi : \mathcal{S} \mapsto \mathcal{A}$ *assigns an action to each state of the Markov decision process. A stochastic stationary* policy $\pi : \mathcal{S} \times \mathcal{A} \mapsto [0,1]$ *satisfies* $\sum_{a \in \mathcal{A}} \pi(s,a) = 1$ *for each* $s \in \mathcal{S}$*. The set of all stochastic stationary policies is denoted as* Π.

Non-stationary policies may take different actions in the same state in different time-steps. We limit our treatment to stationary policies, since for infinite-horizon MDPs there exists an optimal *stationary* and *deterministic* policy. We also consider stochastic policies because they are more convenient to use in some settings. A policy $\pi \in \Pi$ together with the transition matrix induces a distribution over the state space $\mathcal{S}$ in every time step resulting in random variables S_t for $t = 0 \ldots \infty$. The return of a policy is then defined as:

$$\rho(\pi, \alpha) = \mathbf{E}_\alpha \left[\sum_{t=0}^{\infty} \sum_{a \in \mathcal{A}} \gamma^t \pi(S_t, a) r(S_t, a) \right] ,$$

where α is the distribution of S_0. Our objective is then $\max_{\pi \in \Pi} \rho(\pi, \alpha)$, for which the optimal solution is some deterministic policy $\pi^\star$.

The transition matrix and reward function for a *deterministic* policy π are defined as:

$$P_\pi : (s, s') \mapsto P(s, \pi(s), s') \quad \text{and} \quad r_\pi : s \mapsto r(s, \pi(s)) .$$

The transition matrix and reward function for a *stochastic* policy π are defined as:

$$P_\pi : (s, s') \mapsto \sum_{a \in \mathcal{A}} \pi(s,a) P(s,a,s') \quad \text{and} \quad r_\pi : s \mapsto \sum_{a \in \mathcal{A}} \pi(s,a) r(s,a) .$$

In addition, we use P_a and r_a to denote these values for a constant policy $\pi(s) = a$ for some $a \in \mathcal{A}$.

The value function $v : \mathcal{S} \to \mathbb{R}$ represents the expected return when starting in a particular state. The set of all value functions is denoted as $\mathcal{V} = \mathbb{R}^{|\mathcal{S}|}$. A value function v_π of a policy π is: $v_\pi = (\mathbf{I} - \gamma P_\pi)^{-1} r_\pi$.

The value function update for a policy π is denoted by L_π, and the Bellman operator is denoted by L and defined as:

$$L_\pi v = \gamma P_\pi v + r_\pi \ , \qquad\qquad Lv = \max_{\pi \in \Pi} L_\pi v \ .$$

The value function update for a stochastic policy π can be written as:

$$(L_\pi v)(s) = \sum_{a \in \mathcal{A}, s' \in \mathcal{S}} \pi(s,a) \left(\gamma P(s,a,s') v(s') + r(s,a) \right) \ .$$

A policy π is *greedy* with respect to a value function v when $L_\pi v = Lv$. The optimal value function $v^\star = v_{\pi^\star}$ satisfies $v^\star = Lv^\star$. The following proposition summarizes an important property of optimal value functions.

Proposition 3 (Section 6.9 in Puterman, 2005) *For any policy $\pi \in \Pi$ the optimal value function is an upper bound on the value function of any policy:*

$$v^\star \geq v_\pi \ .$$

We assume a vector representation of the policy $\pi \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$. The variables π are defined for all state-action pairs and represent policies. That is, $\pi(s,a)$ represents the probability of taking action $a \in \mathcal{A}$ in state $s \in \mathcal{S}$. The space of all stochastic policies can be represented using the following set of linear equations:

$$\sum_{a \in \mathcal{A}} \pi(s,a) = 1 \qquad\qquad \forall s \in \mathcal{S} \ ,$$
$$\pi(s,a) \geq 0 \qquad\qquad \forall s \in \mathcal{S}, \forall a \in \mathcal{A} \ .$$

These inequalities can be represented using matrix notation as follows:

$$B\pi = \mathbf{1} \qquad\qquad \pi \geq \mathbf{0} \ ,$$

where the matrix $B : |\mathcal{S}| \times (|\mathcal{S}| \cdot |\mathcal{A}|)$ is defined as follows:

$$B(s', (s,a)) = \begin{cases} 1 & s = s' \\ 0 & \text{otherwise} \end{cases} \ .$$

We use $\mathbf{0}$ and $\mathbf{1}$ to denote vectors of all zeros or ones of the appropriate size respectively. The symbol $\mathbf{I}$ denotes an identity matrix of the appropriate dimension.

In addition, a policy π induces a *state occupancy frequency* $u_\pi : \mathcal{S} \to \mathbb{R}$, defined as follows:

$$u_\pi = \left(\mathbf{I} - \gamma P_\pi^\mathsf{T} \right)^{-1} \alpha \ .$$

The set of all occupancy frequencies is denoted as $\mathcal{U} \subseteq \mathbb{R}^{|\mathcal{S}|}$. The return of a policy depends on the state-action occupancy frequencies and $\alpha^\mathsf{T} v_\pi = r_\pi^\mathsf{T} u_\pi$. The optimal state-action occupancy frequency

is $u_{\pi^\star}$ and is often denoted as $u^\star$. *State-action occupancy frequency* $u : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is defined for all states and actions; notice the missing subscript. We use $u|_a : \mathcal{S} \to \mathbb{R}$ to denote the restriction of u to action $a \in \mathcal{A}$ and use $u|_\pi$ equivalently for a deterministic policy π as $u|_\pi : s \mapsto u(s, \pi(s,a))$. State-action occupancy frequencies u must satisfy (e.g., Section 6.9 in Puterman, 2005):

$$\sum_{a \in \mathcal{A}} (\mathbf{I} - \gamma P_a)^\mathsf{T} u|_a = \alpha \qquad \forall a \in \mathcal{A} \; .$$

To formulate approximate linear and bilinear programs, it is necessary to restrict the value functions so that their Bellman residuals are non-negative (or at least bounded from below). We call such value functions transitive-feasible and define them as follows.

Definition 4 *A value function is* transitive-feasible *when $v \geq Lv$. The set of transitive-feasible value functions is:*

$$\mathcal{K} = \{v \in \mathcal{V} \,|\, v \geq Lv\} \; .$$

Given some $\varepsilon \geq 0$, the set of ε-transitive-feasible *value functions is:*

$$\mathcal{K}(\varepsilon) = \{v \in \mathcal{V} \,|\, v \geq Lv - \varepsilon \mathbf{1}\} \; .$$

Notice that the optimal value function $v^\star$ is transitive-feasible.

Next, we summarize the key properties of value functions and policies that we use to derive the results. First, the following lemma summarizes the monotonicity of transition matrices; it follows from the geometric sequence representation of the matrix inverse.

Lemma 5 *[Monotonicity] Let P be a stochastic matrix. Then both linear operators P and $(\mathbf{I} - \gamma P)^{-1}$ are monotonous:*

$$x \geq y \Rightarrow Px \geq Py \; ,$$
$$x \geq y \Rightarrow (\mathbf{I} - \gamma P)^{-1} x \geq (\mathbf{I} - \gamma P)^{-1} y$$

for all x and y.

An important property, which we rely on, is that greedy policies are not affected by adding or subtracting a constant from a value function; we state this well-known property without proof.

Proposition 6 *Let $v \in \mathcal{V}$ be any value function and assume an arbitrary $c \in \mathbb{R}$. Then:*

$$L(v + c\mathbf{1}) = Lv + \gamma c \mathbf{1} \; .$$

In addition, if π is a greedy policy with respect to v it is also greedy with respect to $v + c\mathbf{1}$.

The models we define also rely on the following basic properties of the Bellman operator.

Lemma 7 *Let u be the state-action occupancy frequency of some policy π. Then:*

$$\mathbf{1}^\mathsf{T} u = 1/(1-\gamma) \; .$$

Proof The lemma follows because:

$$\begin{aligned}
\sum_{a\in\mathcal{A}} (u|_a)^\mathsf{T}(\mathbf{I}-\gamma P_a) &= \alpha^\mathsf{T} \ , \\
\sum_{a\in\mathcal{A}} (u|_a)^\mathsf{T}(\mathbf{I}-\gamma P_a)\mathbf{1} &= \alpha^\mathsf{T}\mathbf{1} \ , \\
(1-\gamma)\sum_{a\in\mathcal{A}} (u|_a)^\mathsf{T}\mathbf{1} = 1 &= (1-\gamma)u^\mathsf{T}\mathbf{1} \ .
\end{aligned}$$

■

Finally, an important property of transitive-feasible value functions is that they represent an upper bound on the optimal value function.

Lemma 8 *Transitive feasible value functions form an upper bound on the optimal value function. If $v \in \mathcal{K}(\varepsilon)$ is an ε-transitive-feasible value function, then:*

$$v \geq v^\star - \varepsilon/(1-\gamma)\mathbf{1} \ .$$

Proof Let $P^\star$ and $r^\star$ be the transition matrix and the reward vector of the optimal policy. Then, using Theorem 5, we get:

$$\begin{aligned}
v &\geq Lv - \varepsilon\mathbf{1} \ , \\
v &\geq \gamma P^\star v + r^\star - \varepsilon\mathbf{1} \ , \\
(\mathbf{I}-\gamma P^\star)v &\geq r^\star - \varepsilon\mathbf{1} \ , \\
v &\geq (\mathbf{I}-\gamma P^\star)^{-1} r^\star - \varepsilon/(1-\gamma) \ .
\end{aligned}$$

■

3. Value Function Approximation

This section describes basic methods for value function approximation used to solve large MDPs. Value function approximation, as its name indicates, only computes an approximate value function $\tilde{v}$ of the MDP. The actual solution of the MDP is then the greedy policy π with respect to this value function $\tilde{v}$. The quality of such a policy can be characterized using its value function v_π in one of the following two ways.

Definition 9 (Policy Loss) *Let π be a policy. The* expected policy loss σ_e *of π is defined as:*

$$\sigma_e(\pi) = \rho(\pi^\star,\alpha) - \rho(\pi,\alpha) = \|v^\star - v_\pi\|_{1,\alpha} = \alpha^\mathsf{T}v^\star - \alpha^\mathsf{T}v_\pi \ ,$$

where $\|x\|_{1,c}$ denotes the weighted L_1 norm: $\|x\|_{1,c} = \sum_i |c(i)x(i)|$.
The robust policy loss σ_r *of π is defined as:*

$$\sigma_r(\pi) = \max_{\{\alpha\geq\mathbf{0}\,|\,\mathbf{1}^\mathsf{T}\alpha=1\}} \rho(\pi^\star,\alpha) - \rho(\pi,\alpha) = \|v^\star - v_\pi\|_\infty = \max_{s\in\mathcal{S}} |v^\star(s) - v_\pi(s)| \ .$$

The expected policy loss captures the total loss of discounted reward when following the policy π instead of the optimal policy, given the initial state distribution. The robust policy loss ignores the initial distribution and, in some sense, measures the difference for the worst-case initial distribution.

A set of state features is a necessary component of value function approximation. These features must be supplied in advance and must capture the essential structure of the problem. The features are defined by mapping each state s to a vector $\phi(s)$ of features. We denote $\phi_i : \mathcal{S} \to \mathbb{R}$ to be a function that maps states to the value of feature i:

$$\phi_i(s) = (\phi(s))_i \ .$$

The desirable properties of the features depend strongly on the algorithm, samples, and attributes of the problem; the tradeoffs are not yet fully understood. The function ϕ_i can be treated as a vector, similarly to the value function v.

Value function approximation methods compute value functions that can be represented using the state features. We call such value functions *representable* and define them below.

Definition 10 *Given a* convex *polyhedral set $\tilde{\mathcal{V}} \subseteq \mathcal{V}$, a value function v is* representable *(in $\tilde{\mathcal{V}}$) if $v \in \tilde{\mathcal{V}}$.*

Many methods that compute a value function based on a given set of features have been developed, such as genetic algorithms and neural networks (Bertsekas and Tsitsiklis, 1996). Most of these methods are extremely hard to analyze, computationally complex, and hard to use. Moreover, these complex methods do not satisfy the convexity assumption in Theorem 10. A simpler and more common method is *linear value function approximation*, in which the value function of each state s is represented as a linear combination of *nonlinear features* $\phi(s)$. Linear value function approximation is easy to apply and analyze.

Linear value function approximation can be expressed in terms of matrices as follows. Let the matrix $\Phi : |\mathcal{S}| \times m$ represent the features for the state-space, where m is the number of features. The rows of the feature matrix Φ, also known as the *basis*, correspond to the features of the states $\phi(s)$. The feature matrix can be defined in one of the following two equivalent ways:

$$\Phi = \begin{pmatrix} - & \phi(s_1)^\mathsf{T} & - \\ - & \phi(s_2)^\mathsf{T} & - \\ & \vdots & \end{pmatrix} , \qquad \Phi = \begin{pmatrix} | & | & \\ \phi_1 & \phi_2 & \dots \\ | & | & \end{pmatrix} .$$

The value function v is then represented as $v = \Phi x$ and the set of representable functions is $\tilde{\mathcal{V}} = \text{colspan}\,(\Phi)$.

The goal of value function approximation is not simply to obtain a good value function $\tilde{v}$ but a policy with a small policy loss. Unfortunately, the policy loss of a greedy policy, as formulated in Theorem 9, depends non-trivially on the approximate value function $\tilde{v}$. Often, the only reliable method of precisely computing the policy loss is to simulate the policy, which can be very costly. The following theorem states the most common bound on the robust policy loss.

Theorem 11 *[Robust Policy Loss, Williams and Baird, 1994] Let π be a greedy policy with respect to a value function $\tilde{v}$. Then:*

$$\|v^\star - v_\pi\|_\infty \leq \frac{2}{1-\gamma} \|\tilde{v} - L\tilde{v}\|_\infty \ .$$

In addition, if $\tilde{v} \in \mathcal{K}$ then:

$$\|v^\star - v_\pi\|_\infty \leq \frac{1}{1-\gamma}\|\tilde{v} - L\tilde{v}\|_\infty \,.$$

The bounds in Theorem 11 are often overly conservative because they ignore the initial distribution and do not apply to the expected policy loss. We propose methods that minimize both the standard bounds in Theorem 11 and new tighter bounds on the expected policy loss in Theorem 12.

We are now ready to derive a new bound on the expected policy loss in its most general form. We show later how this bound relates to existing bounds and discuss its properties and special cases.

Theorem 12 *[Expected Policy Loss] Let $\pi \in \Pi$ be a greedy policy with respect to a value function $\tilde{v} \in \mathcal{V}$ and let the state occupancy frequencies of π be bounded as $\underline{u} \leq u_\pi \leq \overline{u}$. Then:*

$$\begin{aligned}\sigma_e(\pi) = \|v^\star - v_\pi\|_{1,\alpha} &= \alpha^\mathsf{T} v^\star - \alpha^\mathsf{T}\tilde{v} + u_\pi^\mathsf{T}(\tilde{v} - L\tilde{v}) \\ &\leq \alpha^\mathsf{T} v^\star - \alpha^\mathsf{T}\tilde{v} + \underline{u}^\mathsf{T}[\tilde{v} - L\tilde{v}]_- + \overline{u}^\mathsf{T}[\tilde{v} - L\tilde{v}]_+ \,,\end{aligned}$$

where $[x]_+ = \max\{x, \mathbf{0}\}$ and $[x]_- = \min\{x, \mathbf{0}\}$ element-wise. In addition, when $\tilde{v} \in \mathcal{K}$, the bound is:

$$\|v^\star - v_\pi\|_{1,\alpha} \leq -\|v^\star - \tilde{v}\|_{1,\alpha} + \|\tilde{v} - L\tilde{v}\|_{1,\overline{u}} \,, \tag{1}$$

$$\|v^\star - v_\pi\|_{1,\alpha} \leq -\|v^\star - \tilde{v}\|_{1,\alpha} + \frac{1}{1-\gamma}\|\tilde{v} - L\tilde{v}\|_\infty \,. \tag{2}$$

Proof Note that:

$$u_\pi^\mathsf{T}(\mathbf{I} - \gamma P_\pi) - \alpha^\mathsf{T} = \mathbf{0}^\mathsf{T} \,, \tag{3}$$

which follows directly from the definition of state-action occupancy frequencies. The bound is then derived as follows:

$$\begin{aligned}\|v^\star - v_\pi\|_\alpha \overset{\text{Theorem 3}}{=} \alpha^\mathsf{T} v^\star - \alpha^\mathsf{T} v_\pi &\overset{(3)}{=} \alpha^\mathsf{T} v^\star - \alpha^\mathsf{T} v_\pi + (u_\pi^\mathsf{T}(\mathbf{I} - \gamma P_\pi) - \alpha^\mathsf{T})\tilde{v} \\ &= \alpha^\mathsf{T} v^\star - r_\pi^\mathsf{T} u_\pi + (u_\pi^\mathsf{T}(\mathbf{I} - \gamma P_\pi) - \alpha^\mathsf{T})\tilde{v} \\ &= \alpha^\mathsf{T} v^\star - r_\pi^\mathsf{T} u_\pi + u_\pi^\mathsf{T}(\mathbf{I} - \gamma P_\pi)\tilde{v} - \alpha^\mathsf{T}\tilde{v} \\ &= \alpha^\mathsf{T} v^\star - \alpha^\mathsf{T}\tilde{v} + u_\pi^\mathsf{T}((\mathbf{I} - \gamma P_\pi)\tilde{v} - r_\pi) \\ &= \alpha^\mathsf{T} v^\star - \alpha^\mathsf{T}\tilde{v} + u_\pi^\mathsf{T}(\tilde{v} - L\tilde{v}) \\ &\leq \alpha^\mathsf{T} v^\star - \alpha^\mathsf{T}\tilde{v} + \underline{u}^\mathsf{T}[\tilde{v} - L\tilde{v}]_- + \overline{u}^\mathsf{T}[\tilde{v} - L\tilde{v}]_+ \,.\end{aligned}$$

Inequality (1) then follows from Theorem 8, which implies that $\tilde{v} \geq v^\star$ and $v \geq Lv$. Inequality (2) follows using the trivial version of Holder's inequality as:

$$\begin{aligned}\alpha^\mathsf{T} v^\star - \alpha^\mathsf{T}\tilde{v} &\overset{\text{Theorem 8}}{=} -\|v^\star - \tilde{v}\|_{1,\alpha} \,, \\ u_\pi^\mathsf{T}(\tilde{v} - L\tilde{v}) &\overset{\text{Holder's}}{\leq} \|u_\pi\|_1 \|\tilde{v} - L\tilde{v}\|_\infty \overset{\text{Theorem 7}}{=} \frac{1}{1-\gamma}\|\tilde{v} - L\tilde{v}\|_\infty \,.\end{aligned}$$

■

Notice that the bounds in Theorem 12 can be minimized even without knowing the optimal $v^\star$. The optimal value function $v^\star$ is independent of the approximate value function $\tilde{v}$ and the greedy policy π depends only on $\tilde{v}$.

Algorithm 1: Approximate policy iteration, where $\mathcal{Z}(\pi)$ denotes a custom value function approximation for the policy π.

$\pi_0, k \leftarrow$ random, 1 ;
while $\pi_k \neq \pi_{k-1}$ **do**
  $\tilde{v}_k \leftarrow \mathcal{Z}(\pi_{k-1})$;
  $\pi_k(s) \leftarrow \arg\max_{a \in \mathcal{A}} r(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s,a,s') \tilde{v}_k(s) \quad \forall s \in \mathcal{S}$;
  $k \leftarrow k+1$;

Remark 13 *Theorem 12 generalizes the bounds established by de Farias (2002, Theorem 3.1), which state that for each $\tilde{v} \in \mathcal{K}$ and a greedy policy π:*

$$\|v^\star - v_\pi\|_{1,\alpha} \leq \frac{1}{1-\gamma} \|v^\star - \tilde{v}\|_{1,(1-\gamma)u_\pi} .$$

This bound is a special case of Inequality (1) *because $\alpha^\mathsf{T} v^\star - \alpha^\mathsf{T} \tilde{v} \leq 0$ and:*

$$\|\tilde{v} - L\tilde{v}\|_{1,u_\pi} \leq \|v^\star - \tilde{v}\|_{1,u_\pi} = \frac{1}{1-\gamma} \|v^\star - \tilde{v}\|_{1,(1-\gamma)u_\pi} ,$$

from $v^\star \leq L\tilde{v} \leq \tilde{v}$.

The methods that we propose require the following standard assumption.

Assumption 1 *All multiples of the constant vector* **1** *are representable in $\tilde{\mathcal{V}}$. That is, $k\mathbf{1} \in \tilde{\mathcal{V}}$ for all $k \in \mathbb{R}$.*

Notice that the representation set $\tilde{\mathcal{V}}$ satisfies Assumption 1 when the first column of Φ is **1**. The impact of including the constant feature is typically negligible because adding a constant to the value function does not change the greedy policy.

Value function approximation algorithms are typically variations of the exact algorithms for solving MDPs. Hence, they can be categorized as approximate value iteration, approximate policy iteration, and approximate linear programming. The ideas behind approximate value iteration can be traced to Bellman (1957), which was followed by many additional research efforts (Bertsekas and Tsitsiklis, 1996; Sutton and Barto, 1998; Powell, 2007). Below, we only discuss approximate policy iteration and approximate linear programming, because they are the methods most closely related to our approach.

Approximate policy iteration (API) is summarized in Algorithm 1. The function $\mathcal{Z}(\pi)$ denotes the specific method used to approximate the value function for the policy π. The two most commonly used methods—*Bellman residual approximation* and *least-squares approximation* (Lagoudakis and Parr, 2003)—minimize the L_2 norm of the Bellman residual.

The approximations based on minimizing L_2 norm of the Bellman residual are common in practice since they are easy to compute and often lead to good results. Most theoretical analyses of API, however, assume minimization of the L_∞ norm of the Bellman residual:

$$\mathcal{Z}(\pi) \in \arg\min_{v \in \tilde{\mathcal{V}}} \|(\mathbf{I} - \gamma P_\pi)v - r_\pi\|_\infty .$$

L_∞-API is shown in Algorithm 1, where $\mathcal{Z}(\pi)$ is calculated by solving the following linear program:

$$\mathcal{Z}(\pi) = \min_{\sigma, v} \left\{ \sigma \,\middle|\, (\mathbf{I} - \gamma P_\pi) v + \mathbf{1}\sigma \geq r_\pi,\ -(\mathbf{I} - \gamma P_\pi) v + \mathbf{1}\sigma \geq -r_\pi, v \in \tilde{\mathcal{V}} \right\} .$$

We are not aware of convergence or divergence proofs of L_∞-API, and such analysis is beyond the scope of this paper. Theoretically, it is also possible to minimize the L_1 norm of the Bellman residual, but we are not aware of any detailed study of such an approximation.

In the above description of API, we assumed that the value function is approximated for all states and actions. This is impossible in practice due to the size of the MDP. Instead, API relies on a subset of states and actions, provided as samples. API is not guaranteed to converge in general and its analysis is typically in terms of limit behavior. The limit bounds are often very loose. We discuss the performance of API and how it relates to approximate bilinear programming in more detail in Section 7.

Approximate linear programming—a method for value function approximation—is based on the linear program formulation of exact MDPs:

$$\begin{array}{ll} \min\limits_{v} & \sum\limits_{s \in \mathcal{S}} c(s) v(s) \\ \text{s.t.} & v(s) - \gamma \sum\limits_{s' \in \mathcal{S}} P(s', s, a) v(s') \geq r(s, a) \quad \forall (s, a) \in (\mathcal{S}, \mathcal{A}) . \end{array} \tag{4}$$

The value c represents a distribution over the states, usually a uniform one. That is, $\sum_{s \in \mathcal{S}} c(s) = 1$. The linear program (4) is often too large to be solved precisely, so it is approximated by assuming that $v \in \tilde{\mathcal{V}}$ (de Farias and van Roy, 2003), yielding the following *approximate linear program*:

$$\begin{array}{ll} \min\limits_{v} & c^\mathsf{T} v \\ \text{s.t.} & Av \geq b , \qquad v \in \tilde{\mathcal{V}} . \end{array} \tag{5}$$

The matrix inequality $Av \geq b$ represents the inequality in (4) and is the following for actions $a_1, a_2, \ldots, a_n \in \mathcal{A}$:

$$\begin{pmatrix} \mathbf{I} - \gamma P_{a_1} \\ \mathbf{I} - \gamma P_{a_2} \\ \vdots \end{pmatrix} = A \quad \geq \quad b = \begin{pmatrix} r_{a_1} \\ r_{a_2} \\ \vdots \end{pmatrix} .$$

The constraint $v \in \tilde{\mathcal{V}}$ denotes the value function approximation. To actually solve this linear program for the simple linear approximation (when $\tilde{\mathcal{V}} = \text{colspan}(\Phi)$), the value function is represented as $v = \Phi x$, which leads to:

$$\begin{array}{ll} \min\limits_{x} & c^\mathsf{T} \Phi x \\ \text{s.t.} & A\Phi x \geq b . \end{array}$$

Appropriate constraints can be added for other choices of $\tilde{\mathcal{V}}$.

Assumption 1 guarantees that (5) is feasible. The following lemma follows directly from the definition of $\mathcal{K}$:

Lemma 14 *A value function v satisfies $Av \geq b$ if and only if $v \in \mathcal{K}$. In addition, if $v \in \mathcal{K}$, then $v \geq v^\star$.*

Theorem 14 implies that an optimal solution $\tilde{v}$ of (5) satisfies: $\tilde{v} \geq v^\star$ from Theorem 8. As a result, the objective of (5) represents the minimization of $\|v - v^\star\|_{1,c} = c^\mathsf{T}(v - v^\star)$ (de Farias, 2002).

Approximate linear programming is guaranteed to converge to a solution and minimize a weighted L_1 norm on the solution quality.

Theorem 15 (Theorem 4.1 in de Farias, 2002) *Given Assumption 1, let $\tilde{v}$ be the solution of* (5)*. If $c = \alpha$ then:*

$$\|v^\star - \tilde{v}\|_{1,\alpha} \leq \frac{2}{1-\gamma} \min_{v \in \tilde{\mathcal{V}}} \|v^\star - v\|_\infty = \frac{2}{1-\gamma} \min_x \|v^\star - \Phi x\|_\infty \ .$$

The difficulty with the solution of ALP is that it is hard to derive guarantees on the policy loss based on the bounds in terms of the L_1 norm; it is possible when the objective function c represents $\bar{u}$, as Theorem 13 shows. In addition, the constant $1/(1-\gamma)$ may be very large when γ is close to 1.

Approximate linear programs are often formulated in terms of samples instead of the full formulation above. The performance guarantees are then based on analyzing the probability that a large number of constraints is violated. It is generally hard to translate the constraint violation bounds to bounds on the quality of the value function and the policy.

4. Bilinear Program Formulations

This section shows how to formulate value function approximation as a separable bilinear program. Bilinear programs are a generalization of linear programs that allows the objective function to include an additional bilinear term. A separable bilinear program consists of two linear programs with independent constraints and is fairly easy to solve and analyze in comparison to non-separable bilinear programs.

Definition 16 (Separable Bilinear Program) *A* separable *bilinear program in the normal form is defined as follows:*

$$\begin{array}{lll} \min\limits_{w,x \,|\, y,z} & s_1^\mathsf{T} w + r_1^\mathsf{T} x + x^\mathsf{T} C y + r_2^\mathsf{T} y + s_2^\mathsf{T} z & \\ s.t. & A_1 x + B_1 w = b_1 \ , & A_2 y + B_2 z = b_2 \ , \\ & w, x \geq \mathbf{0} \ , & y, z \geq \mathbf{0} \ . \end{array} \tag{6}$$

The objective of the bilinear program (6) is denoted as $f(w,x,y,z)$. We separate the variables using a vertical line and the constraints using different columns to emphasize the separable nature of the bilinear program. In this paper, we only use *separable* bilinear programs and refer to them simply as bilinear programs.

The goal in approximate dynamic programming and value function approximation is to find a policy that is close to optimal. The set of acceptable policies is typically restricted to be greedy with respect to *representable* value functions. We define this set of policies $\tilde{\Pi} \subseteq \Pi$ as:

$$\tilde{\Pi} = \{\pi \in \Pi \,\big|\, L_\pi v = Lv, \ v \in \tilde{\mathcal{V}}\} \ .$$

We propose approximate bilinear formulations that minimize the following bounds on robust and expected policy loss.

1. *Robust policy loss*: Minimize $\|v^\star - v_\pi\|_\infty$ by minimizing the bounds in Theorem 11:

$$\min_{\pi \in \tilde{\Pi}} \|v^\star - v_\pi\|_\infty \leq \min_{v \in \tilde{\mathcal{V}}} \frac{1}{1-\gamma} \|v - Lv\|_\infty \ .$$

2. *Expected policy loss*: Minimize $\|v^\star - v_\pi\|_{1,\alpha}$ by minimizing the bounds in Theorem 12:

$$\min_{\pi\in\tilde{\Pi}} \|v^\star - v_\pi\|_{1,\alpha} \leq \alpha^\mathsf{T} v^\star + \min_{v\in\tilde{\mathcal{V}}\cap\mathcal{K}} \left(-\alpha^\mathsf{T}\tilde{v} + \frac{1}{1-\gamma}\|v - Lv\|_\infty \right) .$$

The appropriateness of each formulation depends on the particular circumstances of the domain. For example, minimizing robust bounds is advantageous when the initial distribution is not known and the performance must be consistent under all circumstances. On the other hand, minimizing expected bounds on the value function is useful when the initial distribution is known.

In the formulations described below, we initially assume that samples of all states and actions are used. This means that the precise version of the operator L is available. When solving large problems, the number of samples is often much smaller, due to either subsampling or reduction based on the structure of the MDP. While sampling in linear programs results simply in removal of constraints, in approximate bilinear programs it also leads to a reduction in the number of certain variables, as described in Section 6.

The formulations below denote the value function approximation generically by $v \in \tilde{\mathcal{V}}$. That restricts the value functions to be representable using the features. Representable value functions v can be replaced by a set of variables x as $v = \Phi x$, which reduces the number of variables to the number of features.

4.1 Robust Policy Loss

The solution of the robust approximate bilinear program minimizes the L_∞ norm of the Bellman residual $\|v - Lv\|_\infty$ over the set of representable and transitive-feasible value functions. This minimization can be formulated as follows.

$$\begin{array}{lll} \min\limits_{\pi|\lambda,\lambda',v} & \pi^\mathsf{T}\lambda + \lambda' & \\ \text{s.t.} & B\pi = \mathbf{1} , & \lambda + \lambda'\mathbf{1} \geq Av - b \geq \mathbf{0} , \\ & \pi \geq \mathbf{0} , & \lambda, \lambda' \geq \mathbf{0} , \\ & & v \in \tilde{\mathcal{V}} . \end{array} \tag{7}$$

All the variables are vectors except λ', which is a scalar. The values A and b are identical to the values in (5). The variables λ correspond to all state-action pairs. These variables represent the Bellman residuals that are being minimized. This formulation offers the following guarantees.

Theorem 17 *Let* $(\tilde{\pi}, \tilde{v}, \tilde{\lambda}, \tilde{\lambda'})$ *be an optimal solution of* (7) *and let*

$$v' = \tilde{v} - \frac{\|\tilde{v} - L\tilde{v}\|_\infty}{2(1-\gamma)}\mathbf{1} .$$

Then:

$$\begin{aligned} \tilde{\pi}^\mathsf{T}\tilde{\lambda} + \tilde{\lambda'} = \|\tilde{v} - L\tilde{v}\|_\infty &= \min_{v\in\mathcal{K}\cap\tilde{\mathcal{V}}} \|v - Lv\|_\infty \\ \|v' - Lv'\|_\infty &= \min_{v\in\tilde{\mathcal{V}}} \|v - Lv\|_\infty \\ &\leq (1+\gamma) \min_{v\in\tilde{\mathcal{V}}} \|v - v^\star\|_\infty . \end{aligned}$$

In addition, there exists an optimal $\tilde{\pi} \in \tilde{\Pi}$.

It is important to note that the theorem states that solving the approximate bilinear program is equivalent to minimization over *all* representable value functions, not only the transitive-feasible ones. This follows by subtracting a constant vector $\mathbf{1}$ from $\tilde{v}$ to balance the lower bounds on the Bellman residual error with the upper ones as Theorem 20 shows. This reduces the Bellman residual by $1/2$ without affecting the policy. Finally, note that whenever $v^\star \in \tilde{\mathcal{V}}$, both ABP and ALP will return the optimal value function $v^\star$. The following corollary follows from Theorem 11 and Theorem 17 applied to v'.

Corollary 18 *For any optimal solution $\tilde{v}$ of* (7)*, the policy loss of the greedy policy $\tilde{\pi}$ is bounded by:*

$$\|v^\star - v_{\tilde{\pi}}\|_\infty = \frac{2}{1-\gamma} \min_{v \in \tilde{\mathcal{V}}} \|v - Lv\|_\infty \ .$$

To prove Theorem 17, we first define the following linear programs.

$$f_1(\pi, v) = \min_{\lambda, \lambda'} \left\{ \pi^\mathsf{T} \lambda + \lambda' \,\middle|\, \mathbf{1}\lambda' + \lambda \geq Av - b, \lambda \geq \mathbf{0} \right\} \ ,$$
$$f_2(v) = \min_{\pi} \{ f_1(\pi, v) \,|\, B\pi = \mathbf{1}, \pi \geq \mathbf{0} \} \ .$$

Assuming that $f^\star$ is the optimal solution of (7), then:

$$f^\star = \min_{\pi \in \Pi, v \in \tilde{\mathcal{V}} \cap \mathcal{K}} f_1(\pi, v) = \min_{v \in \tilde{\mathcal{V}} \cap \mathcal{K}} f_2(v) \ .$$

Lemma 19 *Let $v \in \mathcal{K}$ be a transitive-feasible value function and let π be a policy. Then:*

$$f_1(\pi, v) \geq \|v - L_\pi v\|_\infty \ , \tag{8}$$
$$f_2(v) = \|v - Lv\|_\infty \ . \tag{9}$$

In addition, inequality (8) *becomes an equality for any* deterministic *policy π, and there is a deterministic optimal policy that satisfies equality* (9)*.*

Proof To prove (8), notice that for all $s \in \mathcal{S}$ we have that $\sum_{a \in \mathcal{A}} \pi(s,a) = 1$ and $\pi(s,a) \geq 0$. Then:

$$\begin{aligned}
f_1(\pi, v) &\overset{(8)}{=} \lambda' + \sum_{s \in \mathcal{S}, a \in \mathcal{A}} \lambda(s,a)\pi(s,a) \\
&\overset{\lambda(s,a) \geq 0}{\geq} \lambda' + \max_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} \lambda(s,a)\pi(s,a) \\
&= \max_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} \pi(s,a)(\lambda' + \lambda(s,a)) \\
&\geq \max_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} \pi(s,a) \sum_{s' \in \mathcal{S}} (\gamma P(s,a,s')v(s') + r(s,a)) \\
&= \|v - L_\pi v\|_\infty \ .
\end{aligned}$$

To show the equality for a deterministic policy, set $\lambda' = \|v - L_\pi v\|_\infty$ and $\lambda(s, \pi(s)) = \mathbf{0}$ with other elements of λ set arbitrarily. This can be readily shown to be an optimal solution.

To prove (9), note again that $v \in \mathcal{K}$, which implies that $v \geq Lv$. Then, using the fact that the policy defines an action for every state, we get:

$$\begin{aligned} f_2(v) &= \min_{\pi\in\Pi} \|v - L_\pi v\|_\infty = \min_{\pi\in\Pi} \max_{s\in\mathcal{S}} (v - L_\pi v)(s) \\ &= \max_{s\in\mathcal{S}} \min_{\pi\in\Pi} (v - L_\pi v)(s) \\ &= \max_{s\in\mathcal{S}} (v - \max_{\pi\in\Pi} L_\pi v)(s) \\ &= \max_{s\in\mathcal{S}} (v - Lv)(s) = \|v - Lv\|_\infty \,. \end{aligned}$$

The existence of an optimal deterministic solution then follows from the existence of a deterministic greedy policy with respect to a value function. ■

Now, we show that restricting the value functions to be transitive feasible is not limiting, because it does not restrict the set of greedy policies that are considered. To do that, we define the following sets:

$$\mathcal{V}_1 = \arg\min_{v\in\tilde{\mathcal{V}}\cap\mathcal{K}} \|v - Lv\|_\infty \,, \qquad\qquad \mathcal{V}_2 = \arg\min_{v\in\tilde{\mathcal{V}}} \|v - Lv\|_\infty \,.$$

Let Π_1 and Π_2 be sets of greedy policies with respect to $\mathcal{V}_1$ and $\mathcal{V}_2$. The sets $\mathcal{V}_1$ and $\mathcal{V}_2$ satisfy the following important property.

Lemma 20 *Given Assumption 1, let $v_1 \in \mathcal{V}_1$ and $v_2 \in \mathcal{V}_2$, we have the following equalities:*

$$\min_{s\in\mathcal{S}} (v_1 - Lv_1)(s) = 0 \,, \qquad\qquad -\min_{s\in\mathcal{S}} (v_2 - Lv_2)(s) = \max_{s\in\mathcal{S}} (v_2 - Lv_2)(s) \,.$$

Then, define:

$$v_1' = v_1 - \frac{\|v_1 - Lv_1\|_\infty}{2(1-\gamma)}\mathbf{1} \,, \qquad\qquad v_2' = v_2 + \frac{\|v_2 - Lv_2\|_\infty}{(1-\gamma)}\mathbf{1} \,.$$

for which the following holds:

$$\min_{s\in\mathcal{S}} (v_2' - Lv_2')(s) = 0 \,, \qquad\qquad -\min_{s\in\mathcal{S}} (v_1' - Lv_1')(s) = \max_{s\in\mathcal{S}} (v_1' - Lv_1')(s) \,.$$

Proof Assume, for the sake of deriving a contradiction, that $\min_{s\in\mathcal{S}} (v_1 - Lv_1)(s) = \varepsilon > 0$. Then, let $\bar{v_1} = v_1 - \varepsilon/(1-\gamma)\mathbf{1} \in \mathcal{K}$ which implies the following by Theorem 6:

$$\begin{aligned} \|\bar{v_1} - L\bar{v_1}\|_\infty &= \|v_1 - Lv_1 - \varepsilon\mathbf{1}\|_\infty = \max_{s\in\mathcal{S}} (v_1 - Lv_1 - \varepsilon\mathbf{1})(s) \\ &= \max_{s\in\mathcal{S}} (v_1 - Lv_1)(s) - \varepsilon = \|v_1 - Lv_1\|_\infty - \varepsilon \\ &< \|v_1 - Lv_1\|_\infty \,. \end{aligned}$$

This contradicts the optimality of v_1. The inequality for v_2 follows similarly. The rest of the lemma is a simple consequence of Theorem 6. ■

We are now ready to show that neither the set of greedy policies considered nor the policy loss bounds are affected by considering only transitive feasible functions in (7).

Proposition 21 *Given Assumption 1, the following holds:*

$$\begin{aligned} \mathcal{V}_1 &= \left\{ v_2 + \frac{\|v_2 - Lv_2\|_\infty}{(1-\gamma)}\mathbf{1} \,\middle|\, v_2 \in \mathcal{V}_2 \right\} , \\ \|v_1 - Lv_1\|_\infty &= 2\|v_2 - Lv_2\|_\infty \qquad \forall v_1 \in \mathcal{V}_1, \forall v_2 \in \mathcal{V}_2 , \\ \Pi_1 &= \Pi_2 . \end{aligned}$$

Proof To show that $\mathcal{V}_1 \subseteq \left\{ v_2 + \frac{\|v_2 - Lv_2\|_\infty}{(1-\gamma)}\mathbf{1} \,\middle|\, v_2 \in \mathcal{V}_2 \right\}$, assume a $v_1 \in \mathcal{V}_1$ and define:

$$v_2 = v_1 - \frac{\|v_1 - Lv_1\|_\infty}{2(1-\gamma)}\mathbf{1} .$$

Note that $v_2 \in \mathcal{V}$ from Assumption 1, and $2\|v_2 - Lv_2\|_\infty = \|v_1 - Lv_1\|_\infty$ from Theorem 20. To show that $v_2 \in \mathcal{V}_2$ by contradiction, assume that there exists $\bar{v}_2 \in \mathcal{V}_2$ such that $\|\bar{v}_2 - L\bar{v}_2\|_\infty < \|v_2 - Lv_2\|_\infty$ and let $\bar{v}_1 = \bar{v}_2 + \frac{\|\bar{v}_2 - L\bar{v}_2\|_\infty}{(1-\gamma)}\mathbf{1}$. Using Theorem 20, we get:

$$\|\bar{v}_1 - L\bar{v}_1\|_\infty = 2\|\bar{v}_2 - L\bar{v}_2\|_\infty < 2\|v_2 - Lv_2\|_\infty = \|v_1 - Lv_1\|_\infty ,$$

which contradicts the optimality of v_1.

The inclusion $\mathcal{V}_1 \supseteq \left\{ v_2 - \frac{\|v - Lv\|_\infty}{2(1-\gamma)}\mathbf{1} \,\middle|\, v_2 \in \mathcal{V}_2 \right\}$ and $\Pi_1 \supseteq \Pi_2$ can be shown similarly. Finally, Theorem 6 implies that $\Pi_1 = \Pi_2$. ■

Proposition 22 *Given Assumption 1, the minimal Bellman residual for a representable value function can be bounded as follows:*

$$\min_{v \in \tilde{\mathcal{V}}} \|Lv - v\|_\infty \leq (1+\gamma) \min_{v \in \tilde{\mathcal{V}}} \|v - v^\star\|_\infty .$$

Proof Assume that $\hat{v}$ minimizes $\min_{v \in \tilde{\mathcal{V}}} \|v - v^\star\|_\infty \leq \varepsilon$. Then:

$$\begin{aligned} v^\star - \varepsilon\mathbf{1} \leq \quad & v \quad & \leq v^\star + \varepsilon\mathbf{1} , \\ Lv^\star - \gamma\varepsilon\mathbf{1} \leq \quad & Lv \quad & \leq Lv^\star + \gamma\varepsilon\mathbf{1} , \\ Lv^\star - \gamma\varepsilon\mathbf{1} - v \leq \quad & Lv - v \quad & \leq Lv^\star + \gamma\varepsilon\mathbf{1} - v , \\ Lv^\star - v^\star - (1+\gamma)\varepsilon\mathbf{1} \leq \quad & Lv - v \quad & \leq Lv^\star - v^\star + (1+\gamma)\varepsilon\mathbf{1} , \\ -(1+\gamma)\varepsilon\mathbf{1} \leq \quad & Lv - v \quad & \leq (1+\gamma)\varepsilon\mathbf{1} . \end{aligned}$$

■

Theorem 17 now easily follows from the results above.

Proof [Proof of Theorem 17] Let $f^\star$ be the optimal objective value of (7). Then we have from Theorem 19 that:

$$f^\star = \min_{\pi \in \Pi, v \in \tilde{\mathcal{V}} \cap \mathcal{K}} f_1(\pi, v) = \min_{v \in \tilde{\mathcal{V}} \cap \mathcal{K}} f_2(v) = \min_{v \in \tilde{\mathcal{V}} \cap \mathcal{K}} \|v - Lv\|_\infty .$$

The properties of v' follow directly from Theorem 21:

$$\tilde{v} \in \mathcal{V}_1 \Rightarrow v' \in \mathcal{V}_2 \Rightarrow \|v' - Lv'\|_\infty = \min_{v \in \tilde{\mathcal{V}}} \|v - Lv\|_\infty .$$

Note that the existence of an optimal deterministic policy in (7) follows from the existence of a deterministic optimal policy in f_2. The bound on the minimal Bellman residual follows from Theorem 22. ■

4.2 Expected Policy Loss

This section describes bilinear programs that minimize bounds on the expected policy loss for a given initial distribution $\|v - Lv\|_{1,\alpha}$. The initial distribution can be used to derive tighter bounds on the policy loss. We describe two formulations. They respectively minimize an L_∞ and a weighted L_1 norm on the Bellman residual.

The expected policy loss can be minimized by solving the following bilinear formulation.

$$\begin{array}{lll} \min\limits_{\pi|\lambda,\lambda',v} & \pi^\mathsf{T}\lambda + \lambda' - (1-\gamma)\alpha^\mathsf{T} v & \\ \text{s.t.} & B\pi = \mathbf{1}\,, & Av - b \geq \mathbf{0}\,, \\ & \pi \geq \mathbf{0}\,, & \lambda + \lambda'\mathbf{1} \geq Av - b\,, \\ & & \lambda, \lambda' \geq \mathbf{0}\,, \\ & & v \in \tilde{\mathcal{V}}\,. \end{array} \tag{10}$$

Notice that this formulation is identical to the bilinear program (7) with the exception of the term $-(1-\gamma)\alpha^\mathsf{T} v$.

Theorem 23 *Given Assumption 1, any optimal solution* $(\tilde{\pi}, \tilde{v}, \tilde{\lambda}, \tilde{\lambda}')$ *of* (10) *satisfies:*

$$\begin{aligned} \frac{1}{1-\gamma}\left(\tilde{\pi}^\mathsf{T}\tilde{\lambda} + \tilde{\lambda}'\right) - \alpha^\mathsf{T}\tilde{v} &= \frac{1}{1-\gamma}\|L\tilde{v} - \tilde{v}\|_\infty - \alpha^\mathsf{T}\tilde{v} \\ &= \min_{v \in \mathcal{K} \cap \tilde{\mathcal{V}}} \left(\frac{1}{1-\gamma}\|Lv - v\|_\infty - \alpha^\mathsf{T} v\right) \\ &= \min_{v \in \tilde{\mathcal{V}}} \left(\frac{1}{1-\gamma}\|Lv - v\|_\infty - \alpha^\mathsf{T} v\right)\,. \end{aligned}$$

In addition, there exists an optimal $\tilde{\pi} \in \tilde{\Pi}$.

The following bound on the policy loss follows using Theorem 12.

Corollary 24 *There exists an optimal solution* $\tilde{\pi}$ *that is greedy with respect to* $\tilde{v}$ *for which the policy loss is bounded by:*

$$\begin{aligned} \|v^\star - v_{\tilde{\pi}}\|_{1,\alpha} &\leq \left(\min_{v \in \tilde{\mathcal{V}} \cap \mathcal{K}} \frac{1}{1-\gamma}\|Lv - v\|_\infty - \|v^\star - v\|_{1,\alpha}\right) \\ &\leq \left(\min_{v \in \tilde{\mathcal{V}}} \frac{1}{1-\gamma}\|Lv - v\|_\infty - \alpha^\mathsf{T}(v - v^\star)\right)\,. \end{aligned}$$

Proof The proof of Theorem 23 is similar to the proof of Theorem 17 with the main difference being in the definition of functions f_1, and f_2:

$$f_1(\pi, v) = \min_{\lambda,\lambda'} \left\{ \pi^\mathsf{T}\lambda + \lambda' - (1-\gamma)\alpha^\mathsf{T} v \,\middle|\, \mathbf{1}\lambda' + \lambda \geq Av - b, \lambda \geq \mathbf{0} \right\} ,$$
$$f_2(v) = \min_{\pi} \{ f_1(\pi, v) \,|\, B\pi = \mathbf{1}, \pi \geq \mathbf{0} \} .$$

The following lemma can be proved identically to Theorem 19.

Lemma 25 *Let $v \in \mathcal{K}$ be a transitive-feasible value function and let π be a policy. Then:*

$$f_1(\pi, v) \geq \|v - L_\pi v\|_\infty - \|v^\star - v\|_{1,\alpha} , \tag{11}$$
$$f_2(v) = \|v - Lv\|_\infty - \|v^\star - v\|_{1,\alpha} . \tag{12}$$

In addition, (11) *holds with an equality for a* deterministic *policy π, and there is a deterministic optimal policy that satisfies* (12)*.*

The fact that optimization over transitive-feasible value functions does not restrict the resulting policies is proved identically to Theorem 21 with sets $\mathcal{V}_1$ and $\mathcal{V}_2$ that satisfy the same equations. Notice also that the objective function of (10) does not change when subtracting a constant for $v' = v - k\mathbf{1}$ and $k \geq 0$:

$$\frac{1}{1-\gamma}\|v' - Lv'\|_\infty - \alpha^\mathsf{T} v' = \frac{1}{1-\gamma}\|v - Lv\|_\infty - \alpha^\mathsf{T} v ,$$

when $-\min_{s\in\mathcal{S}}(v' - Lv')(s) = \max_{s\in\mathcal{S}}(v' - Lv')(s)$ and $\min_{s\in\mathcal{S}}(v - Lv)(s) = 0$. ■

5. Solving Bilinear Programs

This section describes methods for solving approximate bilinear programs. Bilinear programs can be easily mapped to other global optimization problems, such as mixed integer linear programs (Horst and Tuy, 1996). We focus on a simple iterative algorithm for solving bilinear programs approximately, which also serves as a basis for many optimal algorithms. In addition, we provide a basic problem-specific mixed integer program formulation.

Solving a bilinear program is an NP-complete problem (Bennett and Mangasarian, 1993). The membership in NP follows from the finite number of basic feasible solutions of the individual linear programs, each of which can be checked in polynomial time. The NP-hardness is shown by a reduction from the SAT problem.

There are two main approaches for solving bilinear programs optimally. In the first approach, a relaxation of the bilinear program is solved. The solution of the relaxed problem represents a lower bound on the optimal solution. The relaxation is then iteratively refined, for example by adding cutting plane constraints, until the solution becomes feasible. This is a common method used to solve integer linear programs. The relaxation of the bilinear program is typically either a linear or semi-definite program (Carpara and Monaci, 2009).

In the second approach, feasible, but suboptimal, solutions of the bilinear program are calculated approximately. The approximate algorithms are usually some variation of Algorithm 2. The bilinear

Algorithm 2: Iterative algorithm for solving (6)

$(x_0, w_0) \leftarrow$ random ;
$(y_0, z_0) \leftarrow \arg\min_{y,z} f(w_0, x_0, y, z)$;
$i \leftarrow 1$;
while $y_{i-1} \neq y_i$ *or* $x_{i-1} \neq x_i$ **do**
    $(y_i, z_i) \leftarrow \arg\min_{\{y,z | A_2 y + B_2 z = b_2 \; y,z \geq \mathbf{0}\}} f(w_{i-1}, x_{i-1}, y, z)$;
    $(x_i, w_i) \leftarrow \arg\min_{\{x,w | A_1 x + B_1 w = b_1 \; x,w \geq \mathbf{0}\}} f(w, x, y_i, z_i)$;
    $i \leftarrow i + 1$
return $f(w_i, x_i, y_i, z_i)$

program formulation is then refined—using concavity cuts (Horst and Tuy, 1996)—to eliminate previously computed feasible solutions and solved again. This procedure can be shown to find the optimal solution by eliminating all suboptimal feasible solutions.

The most common and simplest approximate algorithm for solving bilinear programs is Algorithm 2. This algorithm is shown for the general bilinear program (6), where $f(w,x,y,z)$ represents the objective function. The minimizations in the algorithm are linear programs which can be easily solved. Interestingly, as we will show in Section 7, Algorithm 2 applied to ABP generalizes a version of API.

While Algorithm 2 is not guaranteed to find an optimal solution, its empirical performance is often remarkably good (Mangasarian, 1995). Its basic properties are summarized by the following proposition.

Proposition 26 (Theorem 2.2 in Bennett and Mangasarian, 1993) *Algorithm 2 is guaranteed to converge, assuming that the linear program solutions are in a vertex of the optimality simplex. In addition, the global optimum is a fixed point of the algorithm, and the objective value monotonically improves during execution.*

The proof is based on the finite count of the basic feasible solutions of the individual linear programs. Because the objective function does not increase in any iteration, the algorithm will eventually converge.

As mentioned above, any separable bilinear program can be also formulated as a mixed integer linear program (Horst and Tuy, 1996). Such formulation is not practical in our setting, because its size grows quadratically with the size of ABP and its linear relaxations were very loose in our experiments. Below, we present a more compact and structured mixed integer linear program formulation, which relies on the property of ABP that there is always a solution with an optimal deterministic policy (see Theorem 17).

We only show the formulation of the robust approximate bilinear program (7); the same approach applies to all other formulations that we propose. To formulate the mixed integer linear program, assume a given upper bound $\tau \in \mathbb{R}$ for the optimal solution $\lambda^\star$ and all $s \in \mathcal{S}$ and $a \in \mathcal{A}$

such that $\tau \geq \lambda^\star(s,a)$. The mixed integer linear program formulation that corresponds to (7) is:

$$\begin{array}{ll} \min\limits_{z,\pi,\lambda,\lambda',v} & \mathbf{1}^\mathsf{T} z + \lambda' \\ \text{s.t.} & z \geq \lambda - \tau(\mathbf{1}-\pi)\ , \\ & B\pi = \mathbf{1}\ , \\ & \lambda + \lambda'\mathbf{1} \geq Av - b \geq \mathbf{0}\ , \\ & \lambda, \lambda' \geq \mathbf{0}\ , \qquad v \in \tilde{\mathcal{V}}\ , \qquad \pi \in \{0,1\}^{|\mathcal{S}||\mathcal{A}|}\ . \end{array} \tag{13}$$

The following theorem states the correctness of this formulation:

Theorem 27 *Let $(\pi_1, \lambda_1, \lambda_1')$ be an optimal (greedy-policy) solution of* (7) *and let $\tau \geq \lambda_1$. Then:*

$$\left(\pi_1, \lambda_1, \lambda_1', z' = \min_{z \geq \lambda_1 - (\tau - \pi_1)} \mathbf{1}^\mathsf{T} z\right)$$

is an optimal solution of (13) *and vice versa. When in addition f_1 and f_2 are the optimal objective values of* (7) *and* (13)*, then $f_1 = f_2$.*

Proof First, we show that $(\pi_1, \lambda_1, \lambda_1', z = \min_{z \geq \lambda_1 - (\tau - \pi_1)} \mathbf{1}^\mathsf{T} z)$ is feasible in (13) and has the same objective value. Since π_1 is a greedy policy (see Theorem 17), then $\pi_1 \in \{0,1\}^{\mathcal{S}\times\mathcal{A}}$. That is π_1 is feasible in (13). Let then:

$$z_2(s,a) = \begin{cases} \lambda(s,a) & \text{if } \pi_1(s,a) = 1 \\ 0 & \text{otherwise} \end{cases}\ .$$

To show that z_2 is feasible in (13), analyze the following two cases:

$$\begin{aligned} \pi(s,a) = 1: &\qquad z_2(s,a) + \tau(s,a)(1 - \pi_1(s,a)) = z_2(s,a) = \lambda_1(s,a)\ , \\ \pi(s,a) = 0: &\qquad z_2(s,a) + \tau(s,a)(1 - \pi_1(s,a)) \geq \tau(s,a) \geq \lambda_1(s,a)\ . \end{aligned}$$

The objective values must then be identical based on a simple algebraic manipulation. The reverse direction—showing that for any solution of (13) there is a solution of (7) with the same objective value—follows similarly. ■

This mixed integer linear program formulation is much simpler than a general MILP formulation of a bilinear program (Horst and Tuy, 1996).

The performance of the proposed solution methods strongly depends on the actual structure of the problem. As usual with NP-hard problems, there is very little understanding of the theoretical properties that could guarantee faster solution methods. Experimental results, however, show that the ABP-specific formulation can solve problems that are orders of magnitude larger than those that can be solved by the general MILP formulation of ABP.

6. Sampling Guarantees

Typically, the number of states in an MDP is too large to be explicitly enumerated, making it hard to solve even when the value function is restricted to be representable. The usual approach is to sample a limited number of states, actions, and their transitions in order to approximately calculate

the value function. This section shows basic properties of the samples that are sufficient to establish solution quality guarantees with incomplete samples. To derive sampling bounds, we assume in this section that the representable value functions are regularized.

First, we formally define samples and then show how to use them to compute a solution. The samples of the simplest type are defined as follows.

Definition 28 One-step simple samples *are defined as:*

$$\tilde{\Sigma} \subseteq \{(s,a,(s_1 \dots s_n), r(s,a)) \,|\, s,s' \in \mathcal{S},\ a \in \mathcal{A}\} \ ,$$

where $s_1 \dots s_n$ are selected i.i.d. from the distribution $P(s,a,\cdot)$.

Note that $\tilde{\Sigma}$ represents an arbitrary subset of states and actions and may or may not be sampled from a distribution. More informative samples include the full distribution $P(s,a,\cdot)$ instead of samples from the distribution. While these samples are often unavailable in practice, they are useful in the theoretical analysis of sampling issues.

Definition 29 One-step samples with expectation *are defined as follows:*

$$\bar{\Sigma} \subseteq \{(s,a,P(s,a,\cdot), r(s,a)) \,|\, s \in \mathcal{S},\ a \in \mathcal{A}\} \ ,$$

where $P(s,a,\cdot)$ is the distribution over the next states.

The membership of a state in the samples is denoted simply as $s \in \tilde{\Sigma}$ or $(s,a) \in \Sigma$ with the remaining variables, such as $r(s,a)$ considered to be available implicitly.

The sampling models may vary significantly in different domains. The focus of this work is on problems with either a fixed set of available samples or a domain model. Therefore, we do not analyze methods for gathering samples. We also do not assume that the samples come from previous executions, but rather from a deliberate sample-gathering process.

The samples are used to approximate the Bellman operator and the set of transitive-feasible value functions as the following definitions describe.

Definition 30 *The* sampled Bellman operator *and the corresponding set of sampled transitive-feasible functions are defined as:*

$$(\bar{L}(v))(\bar{s}) = \max_{\{a \,|\, (\bar{s},a) \in \bar{\Sigma}\}} r(\bar{s},a) + \gamma \sum_{s' \in \mathcal{S}} P(\bar{s},a,s') v(s') \qquad \forall \bar{s} \in \bar{\Sigma}$$
$$\bar{\mathcal{K}} = \left\{ v \,\middle|\, (\bar{s},a,P(\bar{s},a),r(\bar{s},a)) \in \bar{\Sigma},\ v(\bar{s}) \geq (\bar{L}v)(\bar{s}) \right\} \ .$$

The less-informative set of samples $\tilde{\Sigma}$ can be used as follows.

Definition 31 *The* estimated Bellman operator *and the corresponding set of estimated transitive-feasible functions are defined as:*

$$(\tilde{L}(v))(\bar{s}) = \max_{\{a \,|\, (\bar{s},a) \in \tilde{\Sigma}\}} r(\bar{s},a) + \gamma \frac{1}{n} \sum_{i=1}^{n} v(s_i) \qquad \forall \bar{s} \in \tilde{\Sigma}$$
$$\tilde{\mathcal{K}} = \left\{ v \,\middle|\, (\bar{s},a,(s_1 \dots s_n),r(\bar{s},a)) \in \tilde{\Sigma},\ v(\bar{s}) \geq (\tilde{L}v)(\bar{s}) \right\} \ .$$

Notice that operators $\tilde{L}$ and $\bar{L}$ map value functions to a subset of all states—only states that are sampled. The values for other states are not defined here; they would be defined in a problem-specific way as, for example, the proof of Theorem 32 shows.

The samples can also be used to create an approximation of the initial distribution, or the distribution of visitation frequencies of a given policy. The estimated initial distribution $\bar{\alpha}$ is defined as:

$$\bar{\alpha}(s) = \begin{cases} \alpha(s) & (s,\cdot,\cdot,\cdot) \in \bar{\Sigma} \\ 0 & \text{otherwise} \end{cases} .$$

Most existing sampling bounds for approximate linear programming focus on bounding the probability that a large number of constraints is violated when assuming a distribution over the constraints (de Farias and van Roy, 2004). The difficulty with this approach is that the number of violated constraints does not easily translate to bounds on the quality of the value function, or the policy. In addition, the constraint distribution assumed in the bounds of de Farias and van Roy (2004) is often somewhat arbitrary with no implication on solution quality.

Our approach, on the other hand, is to define properties of the sampled operators that guarantee that the sampling error bounds are small. These bounds do not rely on distributions over constraints and transform directly to bounds on the policy loss. To define bounds on the sampling behavior, we propose the following assumptions. The first assumption limits the error due to missing transitions in the sampled Bellman operator $\bar{L}$.

Assumption 2 (Constraint Sampling Behavior) *There exists $\varepsilon_p \geq 0$ such that for all $v \in \tilde{\mathcal{V}}$:*

$$Lv - \varepsilon_p \mathbf{1} \leq \bar{L}v \leq Lv .$$

Notice that Assumption 2 implies that:

$$\mathcal{K} \subseteq \bar{\mathcal{K}} \subseteq \mathcal{K}(\varepsilon_p) .$$

The second assumption quantifies the error on the estimation of the transitions of the estimated Bellman operator $\tilde{L}$.

Assumption 3 (Constraint Estimation Behavior) *There exists $\varepsilon_s \geq 0$ such that for all $v \in \tilde{\mathcal{V}}$:*

$$Lv - \varepsilon_s \mathbf{1} \leq \tilde{L}v \leq Lv + \varepsilon_s \mathbf{1} .$$

Notice that Assumption 3 implies that:

$$\bar{\mathcal{K}}(-\varepsilon_s) \subseteq \tilde{\mathcal{K}} \subseteq \bar{\mathcal{K}}(\varepsilon_s) .$$

Assumptions 2 and 3 are intentionally generic, so that they apply to a wide range of scenarios. They can be easily satisfied, for example, by making the following Lipschitz continuity assumptions on state features, transitions and rewards.

Assumption 4 *Let $k : \mathcal{S} \to \mathbb{R}^n$ be a map of the state-space to a normed vector space. Then for all $s_1, s_2, s_3 \in \mathcal{S}$ and all features (columns) $\phi_i \in \Phi$, we define K_r, K_P, and K_ϕ such that*

$$\begin{aligned} |r(s_1) - r(s_2)| &\leq K_r \, \|k(s_1) - k(s_2)\| , \\ |p(s_3|s_1,a) - p(s_3|s_2,a)| &\leq K_P \, \|k(s_1) - k(s_2)\| \quad \forall a \in \mathcal{A} , \\ |\phi_i(s_1) - \phi_i(s_2)| &\leq K_\phi \, \|k(s_1) - k(s_2)\| . \end{aligned}$$

This assumption can be used to provide bounds on the sampling error as follows; for more details on tighter and more general bounds see Petrik (2010).

Proposition 32 *Given Assumptions 1 and 4 and that* $\tilde{\mathcal{V}} = \{\Phi x + l\mathbf{1} \,|\, \|x\|_1 \leq \psi, l \in \mathbb{R}\}$*, then Assumption 2 holds with:*

$$\varepsilon_p = \left(K_r + \psi(K_\phi + \gamma K_P)\right) \max_{s \in \mathcal{S}} \min_{\bar{s} \in \bar{\Sigma}} \|k(s) - k(\bar{s})\| \ .$$

Note that $\tilde{\mathcal{V}}$ as defined in Theorem 32 satisfies Assumption 1.

Proof Assume that there exists a constant q such that:

$$\max_{s \in \mathcal{S}} \min_{\bar{s} \in \bar{\Sigma}} \|k(s) - k(\bar{s})\| \leq q \ .$$

Also, define a function $\chi : \mathcal{S} \to \bar{\mathcal{S}}$ that maps each state to the closest sample as follows:

$$\chi(s) = \arg\min_{\bar{s} \in \bar{\Sigma}} \|k(s) - k(\bar{s})\| \ .$$

We will use the following simple extension of Holder's inequality to prove the proposition.

Lemma 33 *The following holds for any* $v \in \tilde{\mathcal{V}} = \{\Phi x + l\mathbf{1} \,|\, \|x\|_1 \leq \psi, l \in \mathbb{R}\}$ *and any* y *such that* $\mathbf{1}^\mathsf{T} y = 0$*.*

$$|y^\mathsf{T} v| \leq |y|^\mathsf{T} |v| \leq \psi \|\Phi y\|_\infty \ .$$

Assumption 4 directly implies the following inequalities:

$$\begin{aligned} \|\phi(\chi(s)) - \phi(s)\|_\infty &\leq q K_\phi \ , \\ |r(\chi(s)) - r(s)| &\leq q K_r \ , \\ \|P(\chi(s), a)^\mathsf{T} \phi_i - P(s,a)^\mathsf{T} \phi_i\|_\infty &\leq q K_p \quad \forall a \in \mathcal{A} \ . \end{aligned}$$

The proposition now follows using simple algebraic manipulation as:

$$\begin{aligned} &\max_{s \in \mathcal{S}} |(v - Lv)(s) - (v - \bar{L}v)(\chi(s))| \\ &\qquad \leq \max_{s \in \mathcal{S}, a \in \mathcal{A}} |(v - \gamma P_a v - r_a)(s) - (v - \gamma P_a v - r_a)(\chi(s))| \\ &\qquad \leq \max_{s \in \mathcal{S}, a \in \mathcal{A}} |\mathbf{1}_s^\mathsf{T} (\Phi x - \gamma P_a \Phi x - r_a) - \mathbf{1}_{\chi(s)}^\mathsf{T} (\Phi x - \gamma P_a \Phi x - r_a)| \\ &\qquad \leq \max_{s \in \mathcal{S}, a \in \mathcal{A}} |(\mathbf{1}_s^\mathsf{T} - \mathbf{1}_{\chi(s)}^\mathsf{T}) \Phi x| + |(\mathbf{1}_s^\mathsf{T} - \mathbf{1}_{\chi(s)}^\mathsf{T}) \gamma P_a \Phi x| + \\ &\qquad\qquad + |(\mathbf{1}_s^\mathsf{T} - \mathbf{1}_{\chi(s)}^\mathsf{T}) r_a| \\ &\qquad \overset{\text{Theorem 33}}{\leq} \max_{s \in \mathcal{S}, a \in \mathcal{A}} \|(\mathbf{1}_s^\mathsf{T} - \mathbf{1}_{\chi(s)}^\mathsf{T}) \Phi\|_\infty \psi + \\ &\qquad\qquad + \|(\mathbf{1}_s^\mathsf{T} - \mathbf{1}_{\chi(s)}^\mathsf{T}) \gamma P_a \Phi\|_\infty \psi + \|(\mathbf{1}_s^\mathsf{T} - \mathbf{1}_{\chi(s)}^\mathsf{T}) r_a\|_\infty \\ &\qquad \leq q K_r + q\psi(K_\phi + \gamma K_p) \ , \end{aligned}$$

where the last inequality follows from Assumption 4. ■

In practice, the estimated Bellman operator is used to formulate the approximate bilinear program. Then, the matrices used in the sampled approximate bilinear program (7) are defined as follows for all $(s_i, a_j) \in \tilde{\Sigma}$.

$$\tilde{A}\Phi = \begin{pmatrix} - & \phi(s_i)^\mathsf{T} - \gamma \frac{1}{m} \sum_{s' \in s'_1 \ldots s'_m} P(s_i, a_j, s') \phi(s')^\mathsf{T} & - \\ - & \vdots & - \end{pmatrix}, \qquad \tilde{b} = \begin{pmatrix} r(s_i, a_j) \\ \vdots \end{pmatrix},$$
$$\tilde{B}(s', (s_i, a_j)) = \mathrm{I}\left\{s' = s_i\right\} \quad \forall s' \in \tilde{\Sigma}\,.$$

The ordering over states in the definitions above is also assumed to be consistent. The sampled version of the bilinear program (7) is then:

$$\begin{array}{lll} \min\limits_{\pi | \lambda, \lambda', x} & \pi^\mathsf{T}\lambda + \lambda' & \\ \text{s.t.} & \tilde{B}\pi = \mathbf{1}\,, & \tilde{A}\Phi x - b \geq \mathbf{0}\,, \\ & \pi \geq \mathbf{0}\,, & \lambda + \lambda' \mathbf{1} \geq \tilde{A}\Phi x - \tilde{b}\,, \\ & & \lambda, \lambda' \geq \mathbf{0}\,. \end{array} \tag{14}$$

The size of the bilinear program (14) scales with the number of samples and features, not with the size of the full MDP, because the variables λ and π are defined only for state-action pairs in $\tilde{\Sigma}$. That is, $|\pi| = |\lambda| = |\{(s,a) \in \Sigma\}|$. The number of constraints in (14) is approximately three times the number of variables λ. Finally, the number of variables x corresponds to the number of approximation features.

Theorem 17 shows that sampled robust ABP minimizes $\|v - \tilde{L}v\|_\infty$ or $\|v - \bar{L}v\|_\infty$. We are now ready to derive sampling bounds on these values that rely on Assumptions 2 and 3 defined above.

Theorem 34 *Let the optimal solutions to the sampled and precise Bellman residual minimization problems be:*

$$v_1 \in \min_{v \in \tilde{\mathcal{V}}} \|v - Lv\|_\infty\,, \qquad v_2 \in \min_{v \in \tilde{\mathcal{V}}} \|v - \bar{L}v\|_\infty\,, \qquad v_3 \in \min_{v \in \tilde{\mathcal{V}}} \|v - \tilde{L}v\|_\infty\,.$$

Value functions v_1, v_2, v_3 correspond to solutions of instances of robust approximate bilinear programs for the given samples. Also let $\hat{v}_i = v_{\pi_i}$, where π_i is greedy with respect to v_i. Then, given Assumptions 1 to 3, the following holds:

$$\|v^\star - \hat{v}_1\|_\infty \leq \frac{2}{1-\gamma} \min_{v \in \tilde{\mathcal{V}}} \|v - Lv\|_\infty\,,$$
$$\|v^\star - \hat{v}_2\|_\infty \leq \frac{2}{1-\gamma} \left(\min_{v \in \tilde{\mathcal{V}}} \|v - Lv\|_\infty + \varepsilon_p \right)\,,$$
$$\|v^\star - \hat{v}_3\|_\infty \leq \frac{2}{1-\gamma} \left(\min_{v \in \tilde{\mathcal{V}}} \|v - Lv\|_\infty + \varepsilon_p + 2\varepsilon_s \right)\,.$$

These bounds show that it is possible to bound policy loss due to incomplete samples. As mentioned above, existing bounds on constraint violation in approximate linear programming (de Farias and van Roy, 2004) typically do not easily lead to policy loss bounds.

Sampling guarantees for other bilinear program formulations are very similar. Because they also rely on an approximation of the initial distribution and the policy loss, they require additional assumptions on the uniformity of state samples.

Proof We show bounds on $\|v_i - Lv_i\|_\infty$; the theorem can then be inferred from Theorem 17, which establishes that ABP minimizes the Bellman residual. The first inequality follows directly from Theorem 17. The second inequality can be derived as:

$$\begin{aligned} v_2 - Lv_2 &\overset{\text{Assumption 2}}{\leq} v_2 - \bar{L}v_2 \\ &\overset{(\star)}{\leq} v_1 - \bar{L}v_1 \\ &\leq v_1 - Lv_1 + \varepsilon_p \mathbf{1} \,. \end{aligned}$$

The third inequality can be derived as:

$$\begin{aligned} v_3 - Lv_3 &\overset{\text{Assumption 2}}{\leq} v_3 - \bar{L}v_3 + \varepsilon_p \mathbf{1} \\ &\overset{\text{Assumption 3}}{\leq} v_3 - \tilde{L}v_3 + \varepsilon_s \mathbf{1} + \varepsilon_p \mathbf{1} \\ &\overset{(\star)}{\leq} v_1 - \tilde{L}v_1 + \varepsilon_s \mathbf{1} + \varepsilon_p \mathbf{1} \\ &\overset{\text{Assumption 3}}{\leq} v_1 - Lv_1 + 2\varepsilon_s \mathbf{1} + \varepsilon_p \mathbf{1} \,. \end{aligned}$$

The star $(\star)$ in the inequalities refers to the fact that $v_i \geq Lv_i$ and that v_i's minimize the corresponding Bellman residuals. ■

To summarize, this section identifies basic assumptions on the sampling behavior and shows that approximate bilinear programming scales well in the face of uncertainty caused by incomplete sampling. More detailed analysis will need to focus on identifying problem-specific assumptions and sampling modes that guarantee the basic conditions, namely satisfying Assumptions 2 and 3. Such analysis is beyond the scope of this paper.

7. Discussion and Related ADP Methods

This section describes connections between approximate bilinear programming and two closely related approximate dynamic programming methods: ALP, and L_∞-API, which are commonly used to solve factored MDPs (Guestrin et al., 2003). Our analysis sheds light on some of their observed properties and leads to a new *convergent* form of approximate policy iteration.

Approximate bilinear programming addresses some important drawbacks of ALP:

1. ALP provides value function bounds with respect to L_1 norm, which does not guarantee small policy loss;
2. ALP's solution quality depends significantly on the heuristically-chosen objective function c in (5) (de Farias, 2002);
3. The performance bounds involve a constant $1/(1-\gamma)$ which can be very large when γ is close to 1; and
4. Incomplete constraint samples in ALP easily lead to unbounded linear programs.

The downside of using approximate bilinear programming is, of course, the higher computational complexity.

The first and the second issues in ALP can be addressed by choosing a problem-specific objective function c (de Farias, 2002). Unfortunately, all existing bounds require that c is chosen based on the optimal ALP solution for c. This is impossible to compute in practice. Heuristic values for c are used instead. Robust approximate bilinear program (7), on the other hand, has no such parameters.

The fourth issue in approximate linear programs arises when the constraints need to be sampled. The ALP may become unbounded with incomplete samples because its objective value is defined using the L_1 norm on the value function, and the constraints are defined using the L_∞ norm of the Bellman residual. In approximate bilinear programs, the Bellman residual is used in both the constraints and objective function. The objective function of ABP is then bounded below by 0 for an arbitrarily small number of samples.

The NP-completeness of ABP compares unfavorably with the polynomial complexity of ALP. However, most other approximate dynamic programming algorithms are not guaranteed to converge to a solution in finite time. As we show below, the exponential time complexity of ABP is unavoidable (unless P = NP).

Proposition 35 (Mangasarian, 1995) *A bilinear program can be solved in NP time.*

The proof is straightforward. There is an optimal solution of the bilinear program such that the solutions of the individual linear programs are basic feasible. The set of all basic feasible solutions is finite, because the feasible regions of w, x and y, z are independent. The value of a basic feasible solution can be calculated in polynomial time.

The following theorem shows that the computational complexity of the ABP formulation is asymptotically the same as the complexity of tightly approximating the value function.

Theorem 36 *Suppose that* $0 < \gamma < 1$ *and* $\varepsilon > 0$. *Then the problem of determining whether the following inequalities hold is NP-complete:*

$$\min_{v \in \mathcal{K} \cap \tilde{\mathcal{V}}} \|Lv - v\|_\infty < \varepsilon \,, \qquad \min_{v \in \tilde{\mathcal{V}}} \|Lv - v\|_\infty < \varepsilon \,.$$

The problem remains NP-complete even when Assumption 1 is satisfied. In addition, it is also NP-complete to determine:

$$\min_{v \in \tilde{\mathcal{V}}} \|Lv - v\|_\infty - \|v^\star - v\|_{1,\alpha} < \varepsilon \,, \qquad \min_{v \in \tilde{\mathcal{V}}} \|Lv - v\|_{1,\bar{u}} - \|v^\star - v\|_{1,\alpha} < \varepsilon \,,$$

assuming that $\bar{u} \geq \mathbf{0}$ *and* $\mathbf{1}^\mathsf{T} \bar{u} = 1$.

As the theorem states, the value function approximation does not become computationally simpler even when Assumption 1 holds—a universal assumption in the paper. Notice that ALP can determine whether $\min_{v \in \mathcal{K} \cap \tilde{\mathcal{V}}} \|Lv - v\|_\infty = 0$ in polynomial time.

Proof The membership in NP follows from Theorem 17 and Theorem 35. We show NP-hardness by a reduction from the 3SAT problem. We first do not make Assumption 1. We show that the theorem holds for $\varepsilon = 1$. The appropriate ε can be obtained by simply scaling the rewards in the MDP.

The main idea is to construct an MDP and an approximation basis, such that the approximation error is small whenever the SAT problem is satisfiable. The values of the states will correspond to

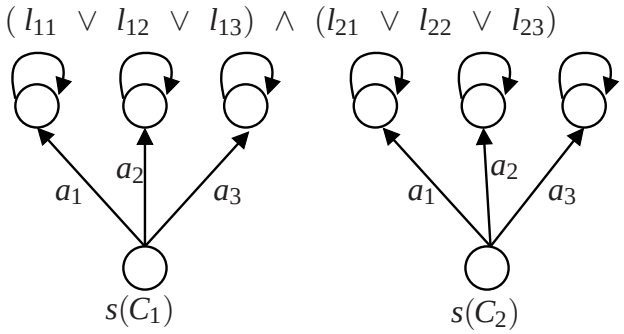

Figure 1: MDP constructed from the corresponding SAT formula.

the truth values of the literals and clauses. The approximation features ϕ will be used to constrain the values of literals that share the same variable. The MDP constructed from the SAT formula is depicted in Figure 1.

Consider a SAT problem with clauses C_i:

$$\bigwedge_{i=1,\ldots,n} C_i = \bigwedge_{i=1,\ldots,n} (l_{i1} \vee l_{i2} \vee l_{i3}) \ ,$$

where l_{ij} are literals. A literal is a variable or the negation of a variable. The variables in the SAT problem are $x_1 \ldots x_m$. The corresponding MDP is constructed as follows. It has one state $s(l_{ij})$ for every literal l_{ij}, one state $s(C_i)$ for each clause C_i and an additional state $\bar{s}$. That is:

$$\mathcal{S} = \{s(C_i) \,|\, i = 1,\ldots,n\} \cup \{s(l_{ij}) \,|\, i = 1,\ldots,n, j = 1,\ldots,3\} \cup \{\bar{s}\} \ .$$

There are 3 actions available in each state $s(C_i)$, which determine the literal of the clause whose value is true. There is only a single action available in states $s(l_{ij})$ and $\bar{s}$. All the MDP's transitions are deterministic. The transition $t(s,a) = (s',r)$ is from the state s to s', when action a is taken, and the reward received is r. The transitions are as follows:

$$\begin{aligned} t(s(C_i), a_j) &= (s(l_{ij}), 1-\gamma) \ , \\ t(s(l_{ij}), a) &= (s(l_{ij}), -(1-\gamma)) \ , \\ t(\bar{s}, a) &= (\bar{s}, 2-\gamma) \ . \end{aligned}$$

Notice that the rewards depend on the discount factor γ, for notational convenience.

There is one approximation feature for every variable x_k such that:

$$\begin{aligned} \phi_k(s(C_i)) &= 0 \ , \\ \phi_k(\bar{s}) &= 0 \ , \\ \phi_k(s(l_{ij})) &= \begin{cases} 1 & \text{if } l_{ij} = x_k \\ -1 & \text{if } l_{ij} = \neg x_k \end{cases} \ . \end{aligned}$$

An additional feature in the problem $\bar{\phi}$ is defined as follows:

$$\begin{aligned} \bar{\phi}(s(C_i)) &= 1 \ , \\ \bar{\phi}(s(l_{ij})) &= 0 \ , \\ \bar{\phi}(\bar{s}) &= 1 \ . \end{aligned}$$

The purpose of state $\bar{s}$ is to ensure that $v(s(c_i)) \geq 2-\gamma$, as we assume in the remainder of the proof.

First, we show that if the SAT problem is satisfiable, then $\min_{v \in \tilde{\mathcal{V}} \cap \mathcal{K}} \|Lv - v\|_\infty < 1$. The value function $\tilde{v} \in \mathcal{K}$ is constructed as a linear sum of the features as: $v = \Phi y$, where $y = (y_1, \ldots, y_m, \bar{y})$. Here y_k corresponds to ϕ_k and $\bar{y}$ corresponds to $\bar{\phi}$. The coefficients y_k are constructed from the truth value of the variables as follows:

$$y_k = \begin{cases} \gamma & \text{if } x_k = \text{true} \\ -\gamma & \text{if } x_k = \text{false} \end{cases},$$
$$\bar{y} = 2-\gamma\,.$$

Now define the *deterministic* policy π as:

$$\pi(s(C_i)) = a_j \text{ where } l_{ij} = \text{true}\ .$$

The true literals are guaranteed to exist from the satisfiability. This policy is greedy with respect to $\tilde{v}$ and satisfies that $\|L_\pi \tilde{v} - \tilde{v}\|_\infty \leq 1-\gamma^2$.

The Bellman residuals for all actions and states, given a value function v, are defined as:

$$v(s) - \gamma v(s') - r\ ,$$

where $t(s,a) = (s', r)$. Given the value function $\tilde{v}$, the residual values are:

$$t(s(C_i), a_j) = (s(l_{ij}), 1-\gamma): \quad \begin{cases} 2-\gamma-\gamma^2+(1-\gamma) = 1-\gamma^2 & \text{if} \quad l_{ij} = \text{true} \\ 2-\gamma+\gamma^2+(1-\gamma) = 1+\gamma^2 & \text{if} \quad l_{ij} = \text{false} \end{cases},$$

$$t(s(l_{ij}), a) = (s(l_{ij}), (1-\gamma)): \quad \begin{cases} \gamma-\gamma^2+1-\gamma = 1-\gamma^2 & \text{if } l_{ij} = \text{true} \\ -\gamma+\gamma^2+1-\gamma = (1-\gamma)^2 > 0 & \text{if } l_{ij} = \text{false} \end{cases},$$

$$t(\bar{s}, a) = (\bar{s}, 1-\gamma): \quad (1-\gamma)+\gamma-1 = 0\ .$$

It is now clear that π is greedy and that:

$$\|L\tilde{v} - \tilde{v}\|_\infty = 1-\gamma^2 < 1\ .$$

We now show that if the SAT problem is not satisfiable then $\min_{v \in \mathcal{K} \cap \tilde{\mathcal{V}}} \|Lv - v\|_\infty \geq 1 - \frac{\gamma^2}{2}$. Now, given a value function v, there are two possible cases for each $v(s(l_{ij}))$: 1) a positive value, and 2) a non-positive value. Two literals that share the same variable will have the same sign, since there is only one feature per each variable.

Assume now that there is a value function $\tilde{v}$. There are two possible cases we analyze: 1) all transitions of a greedy policy are to states with positive value, and 2) there is at least one transition to a state with a non-positive value. In the first case, we have that

$$\forall i\, \exists j,\ \tilde{v}(s(l_{ij})) > 0\ .$$

That is, there is a function $q(i)$, which returns the positive literal for the clause j. Now, create a satisfiable assignment of the SAT problem as follows:

$$x_k = \begin{cases} \text{true} & \text{if } l_{iq(i)} = x_k \\ \text{false} & \text{if } l_{iq(i)} = \neg x_k \end{cases},$$

with other variables assigned arbitrary values. Given this assignment, all literals with states that have a positive value will be also positive. Since every clause contains at least one positive literal, the SAT is satisfiable, which is a contradiction with the assumption. Therefore, there is at least one transition to a state with a non-positive value.

Let C_1 represent the clause with a transition to a literal l_{11} with a non-positive value, without loss of generality. The Bellman residuals at the transitions from these states will be:

$$b_1 = \tilde{v}(s(l_{11})) - \gamma\tilde{v}(s(l_{11})) + (1-\gamma) \geq 0 - 0 + (1-\gamma) = 1-\gamma\,,$$
$$b_1 = \tilde{v}(s(C_1)) - \gamma\tilde{v}(s(l_{11})) - (1-\gamma) \geq 2 - \gamma - 0 - 1 + \gamma = 1\,.$$

Therefore, the Bellman residual $\tilde{v}$ is bounded as:

$$\|L\tilde{v} - \tilde{v}\|_\infty \geq \max\{b_1, b_2\} \geq 1\,.$$

Since we did not make any assumptions on $\tilde{v}$, the claim holds for all representable and transitive-feasible value functions. Therefore, $\min_{v \in \tilde{\mathcal{V}} \cap \mathcal{K}} \|Lv - v\|_\infty \leq 1-\gamma^2$ is and only if the 3SAT problem is feasible.

We now show that the problem remains NP-complete even when Assumption 1 holds. Define a new state $\bar{s_1}$ with the following transition:

$$t(\bar{s_2}, a) = (\bar{s_2}, -\frac{\gamma}{2})\,.$$

All previously introduced features ϕ are zero on the new state. That is $\phi_k(\bar{s_1}) = \bar{\phi}(\bar{s_1}) = 0$. The new constant feature is: $\hat{\phi}(s) = 1$ for all states $s \in \mathcal{S}$, and the matching coefficient is denoted as $\bar{y_1}$. When the formula is satisfiable, then clearly $\min_{v \in \tilde{\mathcal{V}} \cap \mathcal{K}} \|Lv - v\|_\infty \leq 1-\gamma^2$ since the basis is now richer and the Bellman error on the new transition is less than $1-\gamma^2$ when $\bar{y_1} = 0$.

Now we show that when the formula is not satisfiable, then:

$$\min_{v \in \tilde{\mathcal{V}} \cap \mathcal{K}} \|Lv - v\|_\infty \geq 1 - \frac{\gamma^2}{2}\,.$$

This can be scaled to an appropriate ε by scaling the rewards. Notice that

$$0 \leq \bar{y_1} \leq \frac{\gamma}{2}\,.$$

When $\bar{y_1} < 0$, the Bellman residual on transitions $s(C_i) \to s(l_{ij})$ may be decreased by increasing $\bar{y_1}$ while adjusting other coefficients to ensure that $v(s(C_i)) = 2-\gamma$. When $\bar{y_1} > \frac{\gamma}{2}$ then the Bellman residual from the state $\bar{s_1}$ is greater than $1 - \frac{\gamma^2}{2}$. Given the bounds on $\bar{y_1}$, the argument for $y_k = 0$ holds and the minimal Bellman residual is achieved when:

$$v(s(C_i)) - \gamma v(s(l_{ij})) - (1-\gamma) = v(s(\bar{s_1})) - \gamma v(s(\bar{s_1})) + \frac{\gamma}{2}\,,$$
$$2 - \gamma - \gamma\bar{y_1} - (1-\gamma) = \bar{y_1} - \gamma\bar{y_1} + \frac{\gamma}{2}\,,$$
$$\bar{y_1} = \frac{\gamma}{2}\,.$$

Therefore, when the SAT problem is unsatisfiable, the Bellman residual is at least $1 - \frac{\gamma^2}{2}$.

The NP-completeness of $\min_{v \in \tilde{\mathcal{V}}} \|Lv - v\|_\infty < \varepsilon$ follows trivially from the fact that transitive-feasibility does not restrict the solution quality. The proof for $\|v - Lv\|_\infty - \alpha^\mathsf{T} v$ is almost identical. The difference is a new state $\hat{s}$, such that $\phi(\hat{s}) = \mathbf{1}$ and $\alpha(\hat{s}) = 1$. In that case $\alpha^\mathsf{T} v = 1$ for all $v \in \tilde{\mathcal{V}}$. The additional term thus has no effect on the optimization.

The proof can be similarly extended to the minimization of $\|v - Lv\|_{1,\bar{u}}$. Define $\bar{u}(C_i) = 1/n$ and $\bar{u}(l_{ij}) = 0$. Then the SAT problem is satisfiable if an only if $\|v - Lv\|_{1,\bar{u}} = 1 - \gamma^2$. Note that $\bar{u}$, as defined above, is not an upper bound on the occupancy frequencies u_π. It is likely that the proof could be extended to cover the case $\bar{u} \geq u_\pi$ by more carefully designing the transitions from C_i. In particular, there needs to be high probability of returning to C_i and $\bar{u}(l_{ij} > 0$. ■

Approximate bilinear programming can also improve on API with L_∞ minimization (L_∞-API for short), which is a popular method for solving factored MDPs (Guestrin et al., 2003). Minimizing the L_∞ approximation error is theoretically preferable, since it is compatible with the existing bounds on policy loss (Guestrin et al., 2003). The bounds on value function approximation in API are typically (Munos, 2003):

$$\limsup_{k \to \infty} \|v^\star - \hat{v}_k\|_\infty \leq \frac{2\gamma}{(1-\gamma)^2} \limsup_{k \to \infty} \|\tilde{v}_k - v_k\|_\infty ,$$

where $\hat{v}_k$ is the value function of policy π_k which is greedy with respect to $\tilde{v}_k$. These bounds are looser than the bounds on solutions of ABP by at least a factor of $1/(1-\gamma)$. Often the difference may be up to $1/(1-\gamma)^2$ since the error $\|\tilde{v}_k - v_k\|_\infty$ may be significantly larger than $\|\tilde{v}_k - L\tilde{v}_k\|_\infty$. Finally, the bounds cannot be easily used, because they only hold in the limit.

We propose *Optimistic Approximate Policy Iteration* (OAPI), a modification of API. OAPI is shown in Algorithm 1, where $\mathcal{Z}(\pi)$ is calculated using the following program:

$$\begin{array}{ll} \min\limits_{\sigma, v} & \sigma \\ \text{s.t.} & Av \geq b \quad (\equiv (\mathbf{I} - \gamma P_a)v \geq r_a \ \ \forall a \in \mathcal{A}) \\ & -(\mathbf{I} - \gamma P_\pi)v + \mathbf{1}\sigma \geq -r_\pi , \\ & v \in \tilde{\mathcal{V}} . \end{array} \tag{15}$$

In fact, OAPI corresponds to Algorithm 2 applied to ABP because the linear program (15) corresponds to (7) with a fixed π. Then, using Theorem 26, we get the following corollary.

Corollary 37 *Optimistic approximate policy iteration converges in finite time. In addition, the Bellman residual of the generated value functions monotonically decreases.*

OAPI differs from L_∞-API in two ways: 1) OAPI constrains the Bellman residuals by 0 from below and by σ from above, and then it minimizes σ. L_∞-API constrains the Bellman residuals by σ from both above and below. 2) OAPI, like API, uses only the current policy for the upper bound on the Bellman residual, but uses *all* the policies for the lower bound on the Bellman residual. Next we show that the optimal solutions of (16) and (17) are closely related.

L_∞-API cannot return an approximate value function that has a lower Bellman residual than ABP, given the optimality of ABP described in Theorem 17. However, even OAPI—an approximate ABP algorithm—is guaranteed to perform comparably to L_∞-API, as the following theorem states.

Theorem 38 *Assume that L_∞-API converges to a policy $\bar{\pi}$ and a value function $\bar{v}$. Then, define:*

$$\vec{v} = \bar{v} + \frac{1}{1-\gamma}\|\bar{v} - L_{\bar{\pi}}\bar{v}\|_\infty \mathbf{1} .$$

The pair $\bar{\pi}$ and $\vec{v}$ is a fixed point of OAPI when ties are broken appropriately.

Notice that while the optimistic and standard policy iterations can converge to the same solutions, the steps in their computation may not be identical. In addition, there may be multiple points of convergence with the solution depending on the initialization.

Proof First, note that the value function optimization in API and OAPI corresponds to the following optimization problems:

$$\min_{v\in\tilde{\mathcal{V}}} \|L_\pi v - v\|_\infty = \min_{\sigma,v} \left\{ \sigma \;\middle|\; \begin{array}{c} (\mathbf{I}-\gamma P_\pi)v + \mathbf{1}\sigma \geq r_\pi \\ -(\mathbf{I}-\gamma P_\pi)v + \mathbf{1}\sigma \geq -r_\pi \end{array}, v \in \tilde{\mathcal{V}} \right\} , \tag{16}$$

$$\min_{v\in\tilde{\mathcal{V}}\cap\mathcal{K}} \|L_\pi v - v\|_\infty = \min_{\sigma,v} \left\{ \sigma \;\middle|\; \begin{array}{c} (\mathbf{I}-\gamma P_a)v \geq r_a \quad \forall a \in \mathcal{A} \\ -(\mathbf{I}-\gamma P_\pi)v + \mathbf{1}\sigma \geq -r_\pi \end{array}, v \in \tilde{\mathcal{V}} \right\} . \tag{17}$$

Given that $\bar{\pi}$ is greedy with respect to $\bar{v}$ and that $\bar{v}$ minimizes the Bellman residual of $\bar{\pi}$, the following equalities hold:

$$\begin{aligned} L_{\bar{\pi}}\bar{v} &\geq L\bar{v} , \\ \|\bar{v} - L_{\bar{\pi}}\bar{v}\|_\infty &\leq \|v - L_{\bar{\pi}}v\|_\infty \quad \forall v \in \tilde{\mathcal{V}} , \\ -\min_{s\in\mathcal{S}}(v - L_\pi v)(s) &= \max_{s\in\mathcal{S}}(v - L_\pi v)(s) . \end{aligned}$$

Then, $\vec{v} \in \mathcal{K}$ from the first and third properties, since $\vec{v} \geq L_{\bar{\pi}}\vec{v} \geq L\vec{v}$. The value function $\vec{v}$ is therefore feasible in OAPI. In addition, we have that $\|\vec{v} - L_{\bar{\pi}}\vec{v}\|_\infty = 2\|\bar{v} - L_{\bar{\pi}}\bar{v}\|_\infty$.

For the policy $\bar{\pi}$ to be a fixed point in OAPI, it needs to minimize the Bellman residual with respect to $\vec{v}$. This is easy to show as follows:

$$\begin{aligned} L_{\bar{\pi}}\bar{v} &\geq L_\pi\bar{v} , \\ \bar{v} - L_{\bar{\pi}}\bar{v} &\leq \bar{v} - L_\pi\bar{v} , \\ \mathbf{0} \leq \vec{v} - L_{\bar{\pi}}\vec{v} &\leq \vec{v} - L_\pi\vec{v} , \\ \|\vec{v} - L_{\bar{\pi}}\vec{v}\|_\infty &\leq \|\vec{v} - L_\pi\vec{v}\|_\infty . \end{aligned}$$

For the value function $\vec{v}$ to be a fixed point in OAPI, it needs to minimize the Bellman residual with respect to all representable and transitive-feasible value functions. To show a contradiction, assume that there exists $v' \in \tilde{\mathcal{V}} \cap \mathcal{K}$ such that for some $\varepsilon > 0$:

$$\|v' - L_{\bar{\pi}}v'\|_\infty \leq \|v' - L_{\bar{\pi}}v'\|_\infty - \varepsilon .$$

Define also a value function z as follows:

$$z = v' - \left(\frac{1}{2(1-\gamma)}\|\bar{v} - L_{\bar{\pi}}\bar{v}\|_\infty + \frac{\varepsilon}{2} \right) \mathbf{1} .$$

We now show that the Bellman residual of z is less than that of $\bar{v}$:

$$\begin{aligned}
& \|v' - L_{\bar{\pi}} v'\|_\infty && \leq \|\vec{v}' - L_{\bar{\pi}} \vec{v}'\|_\infty - \varepsilon \,, \\
0 \leq \quad & \max_{s \in \mathcal{S}} (v' - L_{\bar{\pi}} v')(s) && \leq \max_{s \in \mathcal{S}} (\vec{v}' - L_{\bar{\pi}} \vec{v}')(s) - \varepsilon \,, \\
-\|\bar{v} - L_{\bar{\pi}} \bar{v}\|_\infty + \frac{\varepsilon}{2} \leq \quad & \max_{s \in \mathcal{S}} (v' - L_{\bar{\pi}} v')(s) - \|\bar{v} - L_{\bar{\pi}} \bar{v}\|_\infty + \tfrac{\varepsilon}{2} && \leq -\|\bar{v} - L_{\bar{\pi}} \bar{v}\|_\infty + \frac{\varepsilon}{2} \,, \\
-\|\bar{v} - L_{\bar{\pi}} \bar{v}\|_\infty + \frac{\varepsilon}{2} \leq \quad & \|z - L_{\bar{\pi}} z\|_\infty && \leq -\|\bar{v} - L_{\bar{\pi}} \bar{v}\|_\infty + \frac{\varepsilon}{2} \,.
\end{aligned}$$

Therefore, $\|z - L_{\bar{\pi}} z\|_\infty < \|\bar{v} - L_{\bar{\pi}} \bar{v}\|_\infty$, which is a contradiction. ■

To summarize, OAPI guarantees convergence, while matching the performance of L_∞-API. The convergence of OAPI is achieved because given a non-negative Bellman residual, the greedy policy also minimizes the Bellman residual. Because OAPI ensures that the Bellman residual is always non-negative, it can progressively reduce it. In comparison, the greedy policy in L_∞-API does not minimize the Bellman residual, and therefore L_∞-API does not always reduce it. Theorem 38 also explains why API provides better solutions than ALP, as observed in Guestrin et al. (2003). From the discussion above, ALP can be seen as an L_1-norm approximation of a single iteration of OAPI. L_∞-API, on the other hand, performs many such ALP-like iterations.

8. Experimental Results

In this section, we validate the approach by applying it to simple reinforcement learning benchmark problems. We consider three different problem domains, each designed to empirically test a different property of the algorithm.

First, in Section 8.1, we compare the policy loss of various approximate bilinear programming formulations with the policy loss of approximate policy iteration and approximate linear programming. These experiments are on a problem that is sufficiently small to compute the optimal value function. Second, in Section 8.2, we compare the solution quality in terms of the Bellman residual for a number of applicable algorithms. Finally, in Section 8.3 we apply ABP with L_1 relaxation to a common inverted pendulum benchmark problem and solve it using the proposed mixed integer linear formulation.

Note that our analysis shows that the solution of ABP using OAPI corresponds to the solutions of API. The optimal solutions of ABP are, therefore, also at least equivalently good in terms of the Bellman residual bounds. However, the actual empirical performance of these methods will depend significantly on the specific problem; our experimental results mostly demonstrate that the proposed methods compute value functions that minimize Bellman residual bounds and result in good policies.

ABP is an off-policy approximation method like LSPI (Lagoudakis and Parr, 2003) or ALP. Thus samples can be gathered independently of the control policy. But it is necessary that multiple actions are sampled for each state to enable the selection of different policies.

8.1 Simple Chain Problem

First, we demonstrate and analyze the properties of ABP on a simple chain problem with 200 states, in which the transitions move to the right or left (2 actions) by one step with a centered Gaussian

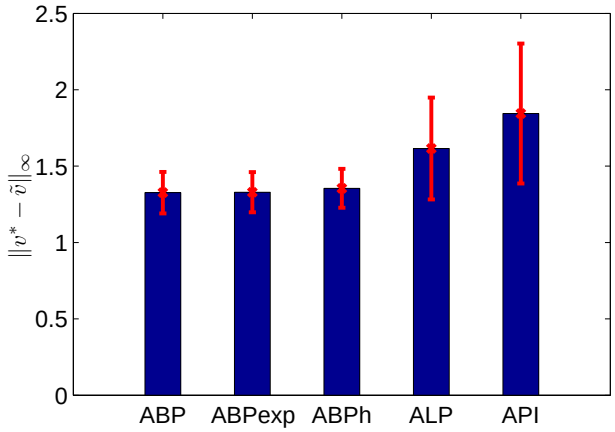

Figure 2: L_∞ Bellman residual for the chain problem

noise of standard deviation 3. The rewards were set to $\sin(i/20)$ for the right action and $\cos(i/20)$ for the left action, where i is the index of the state. This problem is small enough to calculate the optimal value function and to control the approximation features. The approximation basis in this problem is represented by piece-wise linear features, of the form $\phi(s_i) = [i-c]_+$, for c from 1 to 200. The discount factor in the experiments was $\gamma = 0.95$ and the initial distribution was $\alpha(130) = 1$. We verified that the solutions of the bilinear programs were always close to optimal, albeit suboptimal.

We experimented with the full state-action sample and randomly chose the features. All results are averages over 50 runs with 15 features. In the results, we use ABP to denote a close-to-optimal solution of robust ABP, ABPexp for the bilinear program (10), and ABPh for a formulation that minimizes the average of ABP and ABPexp. API denotes approximate policy iteration that minimizes the L_2 norm.

Figure 2 shows the Bellman residual attained by the methods. It clearly shows that the robust bilinear formulation most reliably minimizes the Bellman residual. The other two bilinear formulations are not much worse. Notice also the higher standard deviation of ALP and API. Figure 3 shows the expected policy loss, as specified in Theorem 9, for the calculated value functions. It confirms that the ABP formulation outperforms the robust formulation, since its explicit objective is to minimize the expected loss. Similarly, Figure 4 shows the robust policy loss. As expected, it confirms the better performance of the robust ABP formulation in this case.

Note that API and ALP may achieve lower policy loss on this particular domain than the ABP formulations, even though their Bellman residual is significantly higher. This is possible because ABP simply minimizes bounds on the policy loss. The analysis of tightness of policy loss bounds is beyond the scope of this paper.

8.2 Mountain Car Benchmark Problem

In the mountain-car benchmark, an underpowered car needs to climb a hill (Sutton and Barto, 1998). To do so, it first needs to back up to an opposite hill to gain sufficient momentum. The car receives a reward of 1 when it climbs the hill. The discount factor in the experiments was $\gamma = 0.99$.

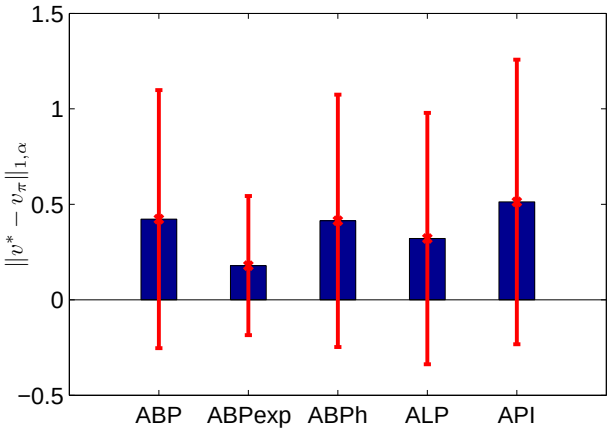

Figure 3: Expected policy loss for the chain problem

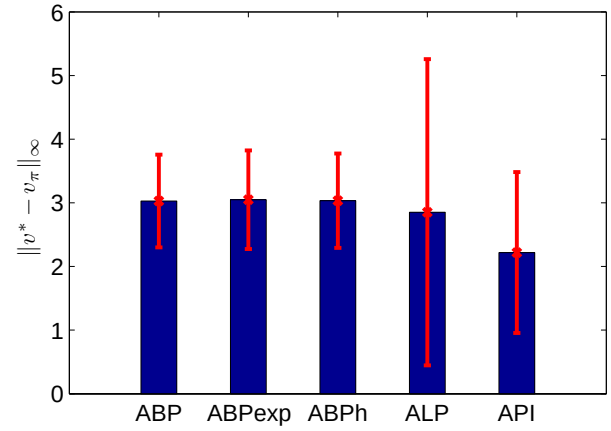

Figure 4: Robust policy loss for the chain problem

Note that the state space in this problem is infinite. It is, therefore, necessary to sample states. The states are sampled uniformly from the feasible state space and the ABP formulation is created as described in Section 6.

The experiments are designed to determine whether OAPI reliably minimizes the Bellman residual in comparison with API and ALP. We use a uniformly-spaced linear spline to approximate the value function. The constraints were based on 200 uniformly sampled states with all 3 actions per state. We evaluated the methods with 100 and 144 approximation features, which correspond to the number of linear segments.

The results of robust ABP (in particular OAPI), ALP, API with L_2 minimization, and LSPI are depicted in Table 1. The results are shown for both L_∞ norm and uniformly-weighted L_2 norm. The run-times of all these methods are comparable, with ALP being the fastest. Since API (LSPI) is not guaranteed to converge, we ran it for at most 20 iterations, which was an upper bound on the number of iterations of OAPI. The results demonstrate that ABP minimizes the L_∞ Bellman residual

(a) L_∞ error of the Bellman residual

Features	100	144
OAPI	0.21 (0.23)	0.13 (0.1)
ALP	13. (13.)	3.6 (4.3)
LSPI	9. (14.)	3.9 (7.7)
API	0.46 (0.08)	0.86 (1.18)

(b) L_2 error of the Bellman residual

Features	100	144
OAPI	0.2 (0.3)	0.1 (1.9)
ALP	9.5 (18.)	0.3 (0.4)
LSPI	1.2 (1.5)	0.9 (0.1)
API	0.04 (0.01)	0.08 (0.08)

Table 1: Bellman residual of the final value function. The values are averages over 5 executions, with the standard deviations shown in parentheses.

much more consistently than the other methods. Note, however, that all the considered algorithms would have performed significantly better with a finer approximation.

8.3 Inverted Pendulum Benchmark Problem

The goal in the inverted pendulum benchmark problem is to balance an inverted pole by accelerating a cart in either of two directions (Wang et al., 1996; Lagoudakis and Parr, 2003). There are three actions in this domain 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 θ and its update equation is:

$$\ddot{\theta} = \frac{g\sin(\theta) - \alpha ml(\dot{\theta})^2\sin(2\theta)/2 - \alpha\cos(\theta)u}{4l/3 - \alpha ml\cos^2(\theta)} .$$

Here the constants are: $g = 9.8$, $m = 2.0$, $M = 8.0$, $\alpha = 1/(m+M)$. The simulation step is set to 0.1 and we use linear interpolation for simplicity.

We used the standard features for this benchmark problem; a set of radial basis functions arranged in a grid over the 2-dimensional state space with centers μ_i and a constant term required by Assumption 1. The features for a state $s = (\theta, \dot{\theta})$ are defined as:

$$\left(1, \exp -\frac{\|s-\mu_1\|_2^2}{2}, \exp -\frac{\|s-\mu_2\|_2^2}{2}, \dots\right) .$$

We considered 100 centers for radial basis functions arranged in a 10 by 10 grid for $\theta \in [-\pi/2, \pi/2]$ and $\dot{\theta} \in [-5, 5]$.

We used L_1 norm regularization to apply the sampling bounds and to compare the approach with regularized approximate linear programming. Assuming that ϕ_0 represents the constant feature, the set of representable value functions is defined as:

$$\tilde{\mathcal{V}} = \left\{ \sum_{i=0}^{100} \phi_i x_i \;\middle|\; \sum_{i=1}^{100} |x_i| \le \psi \right\} .$$

Note that the constant feature is not included in the regularization. The regularization bound was set apriori to $\psi = 100$. Subsequent tests showed that ABP performed almost identically with the regularization bound for values $\psi \in [50, 200]$.

Transition samples were collected in advance—using the same procedure as LSPI—from random episodes, starting in randomly perturbed states very close to the equilibrium state $(0, 0)$ and

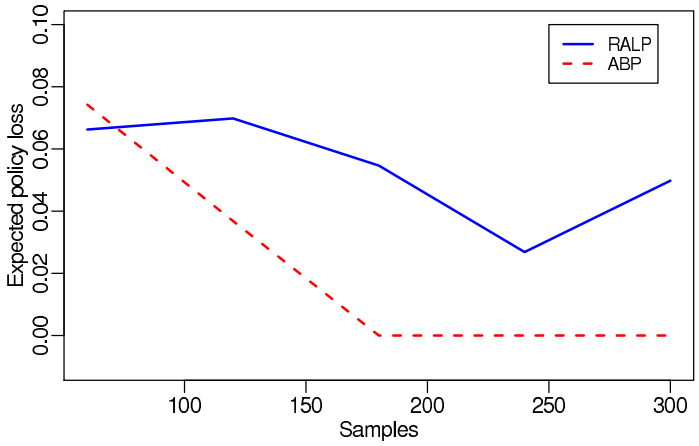

Figure 5: Policy loss as a function of the number of samples.

following a random policy. The average length of such episodes was about 6 steps. We computed the transitions for each sampled state and all the actions by sampling each transition 20 times.

We compare the solution quality to regularized approximate linear programming (RALP), which has been show to perform well on a range of benchmarks (Petrik et al., 2010). We evaluated only the formulation that minimizes the robust objective. The mixed integer linear program formulation for ABP was optimized using CPLEX 12.1. We set the time cutoff to be 60s. In this time interval, most solutions were computed to about 10% optimality gap.

Figure 5 compares the expected policy loss of ABP and RALP on the inverted pendulum benchmark as a function of the number of state transitions sampled. In every iteration, both ABP and RALP were run with the same samples. The policy loss was evaluated on 50 episodes, each at most 50 steps long. The performance of the optimal policy was assumed to be 0 and the policy loss of 0 essentially corresponds to balancing the pole for 2500 steps.

The experimental results on the inverted pendulum demonstrate that ABP may significantly outperform RALP. Both RALP and ABP have a large sampling error when the number of samples is small. This could be addressed by appropriately setting the regularization bound as our sampling bounds indicate; we kept the regularization bound fixed for all sample counts for the sake of simplicity. With a larger number of samples, ABP significantly outperforms RALP, which significantly outperforms LSPI for similar features (Petrik et al., 2010).

9. Conclusion and Future Work

We propose and analyze approximate bilinear programming, a new value-function approximation method, which provably minimizes bounds on policy loss. ABP returns the *optimal* approximate value function with respect to the Bellman residual bounds, despite being formulated with regard to transitive-feasible value functions. We also show that there is no asymptotically simpler formulation, since finding the closest value function and solving a bilinear program are both NP-complete problems. Finally, the formulation leads to the development of OAPI, a new convergent form of API which monotonically improves the objective value function.

While we only discuss simple solvers for ABP, a deeper study of bilinear solvers may lead to more efficient optimal solution methods. ABPs have a small number of essential variables (that

determine the value function) and a large number of constraints, which can be leveraged by some solvers (Petrik and Zilberstein, 2007). In addition, the L_∞ error bound provides good theoretical guarantees, but it may be too conservative in practice; a similar formulation based on L_2 norm minimization may be more practical.

Note that, as for example LSPI, approximate bilinear programming is especially applicable to MDPs with discrete (and small) action spaces. This requirement is limiting in solving many resource management problems in which the resource is a continuous variable. While it is always possible to discretize the action space, this is not feasible when the action space is multidimensional. Therefore, extending these methods to problems with continuous action spaces is an important issue that needs to be addressed in future work.

We believe that the proposed formulation will help deepen the understanding of value function approximation and the characteristics of existing solution methods, and potentially lead to the development of more robust and more widely-applicable reinforcement learning algorithms.

Acknowledgments

This work was supported by the Air Force Office of Scientific Research under Grant No. FA9550-08-1-0171. We also thank the anonymous reviewers for their comments that helped to improve the paper significantly.

References

Pieter Abbeel, Varun Ganapathi, and Andrew Y. Ng. Learning vehicular dynamics, with application to modeling helicopters. In *Advances in Neural Information Processing Systems*, pages 1–8, 2006.

Daniel Adelman. A price-directed approach to stochastic inventory/routing. *Operations Research*, 52:499–514, 2004.

Richard Bellman. *Dynamic Programming*. Princeton University Press, 1957.

Kristin P. Bennett and O. L. Mangasarian. Bilinear separation of two sets in n-space. *Computation Optimization and Applications*, 2, 1993.

Dimitri P. Bertsekas and Sergey Ioffe. Temporal differences-based policy iteration and applications in neuro-dynamic programming. Technical Report LIDS-P-2349, LIDS, 1997.

Dimitri P. Bertsekas and John N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1996.

Alberto Carpara and Michele Monaci. Bidimensional packing by bilinear programming. *Mathematical Programming Series A*, 118:75–108, 2009.

Daniela P. de Farias. *The Linear Programming Approach to Approximate Dynamic Programming: Theory and Application*. PhD thesis, Stanford University, 2002.

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

Daniela P. de Farias and Benjamin van Roy. On constraint sampling in the linear programming approach to approximate dynamic programming. *Mathematics of Operations Research*, 29(3): 462–478, 2004.

Carlos Guestrin, Daphne Koller, Ronald Parr, and Shobha Venkataraman. Efficient solution algorithms for factored MDPs. *Journal of Artificial Intelligence Research*, 19:399–468, 2003.

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

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

Olvi L. Mangasarian. The linear complementarity problem as a separable bilinear program. *Journal of Global Optimization*, 12:1–7, 1995.

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

Marek Petrik. *Optimization-based Approximate Dynamic Programming*. PhD thesis, University of Massachusetts Amherst, 2010.

Marek Petrik and Shlomo Zilberstein. Anytime coordination using separable bilinear programs. In *Conference on Artificial Intelligence*, pages 750–755, 2007.

Marek Petrik and Shlomo Zilberstein. Constraint relaxation in approximate linear programs. In *International Conference on Machine Learning*, pages 809–816, 2009.

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

Warren B. Powell. *Approximate Dynamic Programming*. Wiley-Interscience, 2007.

Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 2005.

Kenneth O. Stanley and Risto Miikkulainen. Competitive coevolution through evolutionary complexification. *Journal of Artificial Intelligence Research*, 21:63–100, 2004.

Richard S. Sutton and Andrew Barto. *Reinforcement Learning*. MIT Press, 1998.

Istvan Szita and Andras Lorincz. Learning Tetris using the noisy cross-entropy method. *Neural Computation*, 18(12):2936–2941, 2006.

Hua O. Wang, Kazuo Tanaka, and Meichael F. Griffin. An approach to fuzzy control of nonlinear systems: Stability and design issues. *IEEE Transactions on Fuzzy Systems*, 4:14–23, 1996.

Ronald J. Williams and Leemon C. Baird. Tight performance bounds on greedy policies based on imperfect value functions. In *Yale Workshop on Adaptive and Learning Systems*, 1994.