June 14, 2010

Global Optimization for Value Function Approximation

Marek Petrik

Shlomo Zilberstein, *University of Massachusetts - Amherst*

Global Optimization for Value Function Approximation

Marek Petrik   PETRIK@CS.UMASS.EDU
Department of Computer Science
University of Massachusetts
Amherst, MA 01003, USA

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

Editor:

Abstract

Existing value function approximation methods have been successfully used in many applications, but they 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 is unavoidable because the Bellman-residual minimization itself is NP-hard. We describe and analyze both optimal and approximate algorithms 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, Markov decision processes, reinforcement learning, approximate dynamic programming

1. Motivation

Solving large Markov Decision Problems (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 only be solved 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*, which searches a restricted space of value functions, and 3) *approximate linear programming*, which approximates the solution using a linear program. While all of these methods have achieved impressive results in many 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^* 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, the 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.

We present new formulations of value function approximation that provably minimize bounds on the policy loss using global optimization. Most of these bounds do not rely on values that are hard to obtain, unlike, for example, approximate linear programming. The focus of the work is on two broad bound minimization approaches: 1) minimizing L_∞ bounds, and 2) minimizing weighted L_1 norm bounds on the policy loss. In some sense, the formulations minimize the bounds by unifying policy value-function search methods.

We start with a description of the framework and notation in Section 2 and the 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 also show in Section 5 that this 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. Technical proofs are provided in the appendix.

The general setting considered in this paper is a restriction of reinforcement learning. 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 formally defines the framework and the notation we use. We also define Markov decision processes and the approximation errors involved. Markov decision processes come in many flavors based on the objective function that is optimized. This work focuses on the infinite horizon discounted MDPs, which are defined as follows.

Definition 1 (e.g. (Puterman, 2005)). A *Markov Decision Process* is a tuple $(\mathcal{S}, \mathcal{A}, P, r, \alpha)$. Here, $\mathcal{S}$ is a finite set of states, $\mathcal{A}$ is a finite set of actions, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0, 1]$ is the transition function ($P(s, a, s')$ is the probability of transiting to state s' from state s given action a), and $r : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}_+$ is a reward function. The initial distribution is: $\alpha : \mathcal{S} \mapsto [0, 1]$, such that $\sum_{s \in \mathcal{S}} \alpha(s) = 1$.

The goal is to find a sequence of actions that maximizes γ-discounted discounted cumulative sum of the rewards, also called the *return*. A solution of a Markov decision process is a policy, which is 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 policy *policy* $\pi : \mathcal{S} \times \mathcal{A} \mapsto [0, 1]$. The set of all stochastic stationary policies is denoted as Π and satisfies $\sum_{a \in \mathcal{A}} \pi(s, a) = 1$.

General non-stationary policies may take different actions in states 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 that we consider.

The transition and reward functions for a given policy are denoted by P_π and r_π. The value function update for a policy π is denoted by L_π, and the Bellman operator is denoted by L. That is:

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

The optimal value function, denoted v^*, satisfies $v^* = Lv^*$.

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 correct (stochastic) policies can be represented using a set of linear equations:

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

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

3

These inequalities can be represented using matrix notation as follows.

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

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 visitation 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 return of a policy depends on the state-action visitation frequencies and $\alpha^{\mathsf{T}} v_\pi = r^{\mathsf{T}} u_\pi$. The optimal state-action visitation frequency is u_{π^*}. *State-action visitation 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 to denote the part of u that corresponds to action $a \in \mathcal{A}$. State-action visitation frequencies must satisfy:

$$\sum_{a \in \mathcal{A}} (\mathbf{I} - \gamma P)^{\mathsf{T}} u_a = \alpha.$$

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

$$\mathcal{K} = \{v \in \mathbb{R}^{|\mathcal{S}|} \,|\, v \geq Lv\}.$$

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

$$\mathcal{K}(\epsilon) = \{v \in \mathbb{R}^{|\mathcal{S}|} \,|\, v \geq Lv - \epsilon\mathbf{1}\}.$$

Notice that the optimal value function v^* is transitive-feasible. The following lemma summarizes the key property of transitive-feasible value functions:

Lemma 4. *Transitive feasible value functions form an upper bound on the optimal value function. If $v \in \mathcal{K}(\epsilon)$ is an ϵ-transitive-feasible value function, then*

$$v \geq v^* - \frac{\epsilon}{1 - \gamma}\mathbf{1}.$$

3. Value Function Approximation

This section describes the basic methods for value function approximation. MDPs used in practical applications are often too large for the optimal policy to be computed precisely. In these cases, we first calculate an approximate value function $\tilde{v}$ and then take the greedy policy π with respect to it. The quality of such a policy can be characterized using its value function v_π in one of the following two main ways.

Definition 5 (Policy Loss). Let π be a policy computed from value function approximation. The *expected policy loss* measures the expected loss of π, defined as follows:

$$\|v^* - v_\pi\|_{1,\alpha} = \alpha^\mathsf{T} v^* - \alpha^\mathsf{T} v_\pi \tag{1}$$

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

$$\|v^* - v_\pi\|_\infty = \max_{s \in \mathcal{S}} |v^*(s) - v_\pi(s)| \tag{2}$$

The expected policy loss captures the total loss of discounted reward when following the policy π instead of the optimal policy assuming the initial 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 also 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 6. Given a *convex* polyhedral set $\mathcal{M} \subseteq \mathbb{R}^{|\mathcal{S}|}$, a value function v is *representable* (in $\mathcal{M}$) if $v \in \mathcal{M}$.

Many methods that compute a value function based on a given set of features have been developed, such as neural networks and genetic algorithms (Bertsekas and Ioffe, 1997). 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 Definition 6. A simpler, more common, method is *linear value function approximation*. In linear value function approximation, the value function of 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

5

features of the states $\phi(s)$. The feature matrix can be defined in one of the following two ways:

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

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

The goal of value function approximation is not just 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 Definition 5, 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. Value function approximation methods, therefore, optimize bounds on the policy loss.

Theorem 7. *[Robust Policy Loss, e.g. (Williams and Baird, 1994)] Let π be a greedy policy with respect to a value function $\tilde{v}$. Then:*

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

In addition, if $\tilde{v} \in \mathcal{K}$, the policy loss is minimized for the greedy policy and:

$$\|v^* - v_\pi\|_\infty \le \frac{1}{1-\gamma}\|\tilde{v} - L\tilde{v}\|_\infty.$$

The bounds above ignore the initial distribution and may often be overly conservative. We establish new bounds on the expected policy loss that also consider the initial distribution.

Theorem 8. *[Expected Policy Loss] Let π be a greedy policy with respect to a value function $\tilde{v}$ and let the state-action visitation frequencies of π be bounded as $\underline{u} \le u_\pi \le \bar{u}$. Then:*

$$\|v^* - v_\pi\|_{1,\alpha} = \alpha^{\mathsf{T}} v^* - \alpha^{\mathsf{T}} \tilde{v} + u_\pi^{\mathsf{T}}\left(\tilde{v} - L\tilde{v}\right)$$
$$\le \alpha^{\mathsf{T}} v^* - \alpha^{\mathsf{T}} \tilde{v} + \underline{u}^{\mathsf{T}}\left[\tilde{v} - L\tilde{v}\right]_- + \bar{u}^{\mathsf{T}}\left[\tilde{v} - L\tilde{v}\right]_+.$$

The state-visitation frequency u_π depends on the initial distribution α, unlike v^*. In addition, when $\tilde{v} \in \mathcal{K}$, the bound is:

$$\|v^* - v_\pi\|_{1,\alpha} \le -\|v^* - \tilde{v}\|_{1,\alpha} + \|\tilde{v} - L\tilde{v}\|_{1,\bar{u}}$$

$$\|v^* - v_\pi\|_{1,\alpha} \le -\|v^* - \tilde{v}\|_{1,\alpha} + \frac{1}{1-\gamma}\|\tilde{v} - L\tilde{v}\|_\infty$$

Here we use $[x]_+ = \max\{x, \mathbf{0}\}$ componentwise.

Proof. The bound is derived as follows:

$$\begin{aligned}
\alpha^{\mathsf{T}} v^* - \alpha^{\mathsf{T}} v_\pi &= \alpha^{\mathsf{T}} v^* - \alpha^{\mathsf{T}} v_\pi + (u_\pi^{\mathsf{T}}(\mathbf{I} - \gamma P_\pi) - \alpha^{\mathsf{T}})\tilde{v} \\
&= \alpha^{\mathsf{T}} v^* - r_\pi^{\mathsf{T}} u_\pi + (u_\pi^{\mathsf{T}}(\mathbf{I} - \gamma P_\pi) - \alpha^{\mathsf{T}})\tilde{v} \\
&= \alpha^{\mathsf{T}} v^* - r_\pi^{\mathsf{T}} u_\pi + u_\pi^{\mathsf{T}}(\mathbf{I} - \gamma P_\pi)\tilde{v} - \alpha^{\mathsf{T}}\tilde{v} \\
&= \alpha^{\mathsf{T}} v^* - \alpha^{\mathsf{T}}\tilde{v} + u_\pi^{\mathsf{T}}\left((\mathbf{I} - \gamma P_\pi)\tilde{v} - r_\pi\right) \\
&= \alpha^{\mathsf{T}} v^* - \alpha^{\mathsf{T}}\tilde{v} + u_\pi^{\mathsf{T}}\left(\tilde{v} - L\tilde{v}\right).
\end{aligned}$$

6

We used the fact that $u_\pi^\mathsf{T}(\mathbf{I} - \gamma P_\pi) - \alpha^\mathsf{T} = \mathbf{0}$, which follows from the definition of state-action visitation frequencies and that $v^* \geq v_\pi$. The inequalities for $\tilde{v} \in \mathcal{K}$ follow from Lemma 38, Lemma 43, and the trivial version of the Holder's inequality:

$$\alpha^\mathsf{T} v^* - \alpha^\mathsf{T} \tilde{v} = -\|v^* - \tilde{v}\|_{1,\alpha}$$

$$u_\pi^\mathsf{T}(\tilde{v} - L\tilde{v}) \leq \|u_\pi\|_1 \|\tilde{v} - L\tilde{v}\|_\infty = \frac{1}{1-\gamma}\|\tilde{v} - L\tilde{v}\|_\infty$$

$\square$

Notice that the bounds in Theorem 8 can be minimized even without knowing the optimal v^*. The optimal value function v^* is independent of the approximate value function $\tilde{v}$ and the greedy policy π depends only on $\tilde{v}$.

Remark 9. The bounds in Theorem 8 generalize the bounds established by de Farias (2002, Theorem 1.3), which state that whenever $v \in \mathcal{K}$:

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

This bound is a special case of Theorem 8 because $\alpha^\mathsf{T} v^* - \alpha^\mathsf{T} \tilde{v} \leq 0$ and:

$$\|\tilde{v} - L\tilde{v}\|_{1,u} \leq \|v^* - \tilde{v}\|_{1,u} \leq \frac{1}{1-\gamma}\|v^* - \tilde{v}\|_{1,(1-\gamma)u},$$

from $v^* \leq L\tilde{v} \leq \tilde{v}$. The proof of Theorem 8 also simplifies the proof of Theorem 1.3 in (de Farias, 2002).

The methods that we propose require the following standard assumption.

Assumption 10. All multiples of the constant vector $\mathbf{1}$ are representable in $\mathcal{M}$. That is, for all $k \in \mathbb{R}$ we have that $k\mathbf{1} \in \mathcal{M}$.

Notice that the representation set $\mathcal{M}$ satisfies Assumption 10 when a first column of Φ is $\mathbf{1}$. 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 of approximate value iteration could 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 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:

$$\mathcal{Z}(\pi) \in \arg\min_{v \in \mathcal{M}} \|(\mathbf{I} - \gamma P_\pi)v - r_\pi\|_\infty \tag{3}$$

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

---

1 $\pi_0, k \leftarrow$ rand, 1 ;
2 while $\pi_k \neq \pi_{k-1}$ **do**
3 $\quad$ $\tilde{v}_k \leftarrow \mathcal{Z}(\pi_{k-1})$;
4 $\quad$ $\pi_k(s) \leftarrow \arg\max_{a \in \mathcal{A}} r(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s,a,s')\tilde{v}_k(s) \quad \forall s \in \mathcal{S}$;
5 $\quad$ $k \leftarrow k + 1$;

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

$$
\begin{aligned}
\min_{\phi, v} \quad & \phi \\
\text{s.t.} \quad & (\mathbf{I} - \gamma P_\pi)v + \mathbf{1}\phi \geq r_\pi \\
& -(\mathbf{I} - \gamma P_\pi)v + \mathbf{1}\phi \geq -r_\pi \\
& v \in \mathcal{M}
\end{aligned}
\tag{4}
$$

We are not aware of convergence or divergence proofs of L_∞-API, and this 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 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 only 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:

$$
\begin{aligned}
\min_{v} \quad & \sum_{s \in \mathcal{S}} c(s)v(s) \\
\text{s.t.} \quad & v(s) - \gamma \sum_{s' \in \mathcal{S}} P(s',s,a)v(s') \geq r(s,a) \quad \forall (s,a) \in (\mathcal{S}, \mathcal{A})
\end{aligned}
\tag{5}
$$

We use A as a shorthand notation for the constraint matrix and b for the right-hand side. 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 (5) is often too large to be solved precisely, so it is approximated to get an *approximate linear program* by assuming that $v \in \mathcal{M}$ (de Farias and van Roy, 2003), as follows:

$$
\begin{aligned}
\min_{v} \quad & c^\mathsf{T} v \\
\text{s.t.} \quad & Av \geq b \\
& v \in \mathcal{M}
\end{aligned}
\tag{ALP–L_1}
$$

The constraint $v \in \mathcal{M}$ denotes the approximation. To actually solve this linear program, the value function is represented as $v = \Phi x$. Assumption 10 guarantees the feasibility of

the ALP. The optimal solution of the ALP, $\tilde{v}$, satisfies: $\tilde{v} \geq v^*$. Therefore, the objective of (ALP–L_1) represents the minimization of $\|\tilde{v} - v^*\|_{1,c}$ (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 11 (e.g. (de Farias, 2002)). *Given Assumption 10, let $\tilde{v}$ be the solution of the approximate linear program* (ALP–L_1). *If $c = \alpha$ then*

$$\|v^* - \tilde{v}\|_{1,\alpha} \leq \frac{2}{1-\gamma} \min_{v \in \mathcal{M}} \|v^* - v\|_\infty.$$

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 Remark 9 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 with an additional bilinear term in the objective function. A separable bilinear program consists of two linear programs with independent constraints and are fairly easy to solve and analyze.

Definition 12 (Separable Bilinear Program). A *separable* bilinear program in the normal form is defined as follows:

$$\begin{aligned} \min_{w,x \,|\, y,z} \quad & s_1^\mathsf{T} w + r_1^\mathsf{T} x + x^\mathsf{T} C y + r_2^\mathsf{T} y + s_2^\mathsf{T} z \\ \text{s.t.} \quad & A_1 x + B_1 w = b_1 \qquad\qquad A_2 y + B_2 z = b_2 \qquad\qquad \text{(BP–m)} \\ & w, x \geq \mathbf{0} \qquad\qquad\qquad\quad\; y, z \geq \mathbf{0} \end{aligned}$$

The objective of the bilinear program (BP–m) 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.

We present three different approximate bilinear formulations that minimize the following bounds on the approximate value function.

1. *Robust policy loss*: Minimizes $\|v^* - v_\pi\|_\infty$ by minimizing the bounds in Theorem 7:

$$\min_{\pi \in \Pi} \|v^* - v_\pi\|_\infty \leq \min_{v \in \mathcal{M}} \frac{1}{1-\gamma} \|v - Lv\|_\infty$$

2. *Expected policy loss*: Minimizes $\|v^* - v)\pi\|_{1,\alpha}$ by minimizing the bounds in Theorem 8:

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

$$\min_{\pi \in \Pi} \|v^* - v_\pi\|_{1,\alpha} \leq \alpha^\mathsf{T} v^* + \min_{v \in \mathcal{M}} \left(-\alpha^\mathsf{T}\tilde{v} + \|v - Lv\|_{\bar{u}} \right).$$

3. *The sum of k largest errors*: This formulation represents a hybrid between the robust and expected formulations. It is more robust than simply minimizing the expected performance but is not as sensitive to worst-case performance.

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. To solve large problems, the number of samples would be much smaller; either simply subsampled or reduced using the structure of the MDP. Reducing the number of constraints in linear programs corresponds to simply removing constraints. In approximate bilinear programs it also reduces the number of some variables, as Section 6 describes.

The formulations below denote the value function approximation generically by $v \in \mathcal{M}$. That restricts the value functions to be representable using features. Representable value functions v can be replaced by a set of variables x as $v = \Phi x$. This 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 $\|v - Lv\|_\infty$. This minimization can be formulated as follows.

$$
\begin{aligned}
\min_{\pi \mid \lambda, \lambda', v} \quad & \pi^\mathsf{T}\lambda + \lambda' \\
\text{s.t.} \quad & B\pi = \mathbf{1} \quad Av - b \geq \mathbf{0} \\
& \pi \geq \mathbf{0} \quad \lambda + \lambda'\mathbf{1} \geq Av - b \\
& \lambda, \lambda' \geq \mathbf{0} \\
& v \in \mathcal{M}
\end{aligned}
\qquad \text{(ABP–}L_\infty\text{)}
$$

All the variables are vectors except λ', which is a scalar. The matrix A represents constraints that are identical to the constraints in (ALP–L_1). 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 13. *Given Assumption 10, any optimal solution $(\tilde{\pi}, \tilde{v}, \tilde{\lambda}, \tilde{\lambda}')$ of the approximate bilinear program* (ABP–L_∞) *satisfies:*

$$
\begin{aligned}
\tilde{\pi}^\mathsf{T}\tilde{\lambda} + \tilde{\lambda}' = \|L\tilde{v} - \tilde{v}\|_\infty &\leq \min_{v \in \mathcal{K} \cap \mathcal{M}} \|Lv - v\|_\infty \\
&\leq 2 \min_{v \in \mathcal{M}} \|Lv - v\|_\infty \\
&\leq 2(1 + \gamma) \min_{v \in \mathcal{M}} \|v - v^*\|_\infty.
\end{aligned}
$$

Moreover, there exists an optimal solution $\tilde{\pi}$ that is greedy with respect to $\tilde{v}$ for which the policy loss is bounded by:

$$\|v^* - v_{\tilde{\pi}}\|_\infty \leq \frac{2}{1 - \gamma} \left(\min_{v \in \mathcal{M}} \|Lv - v\|_\infty \right).$$

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. This reduces the Bellman residual by $1/2$ without affecting the policy. Finally, note that whenever $v^* \in \mathcal{M}$, both ABP and ALP will return the optimal value function v^*.

To prove the theorem, we first define the following linear program, which solves for the L_∞ norm of the Bellman update L_π for fixed value function v and policy π.

$$\begin{aligned}
f_1(\pi, v) = \min_{\lambda, \lambda'} \quad & \pi^\mathsf{T} \lambda + \lambda' \\
\text{s.t.} \quad & \mathbf{1}\lambda' + \lambda \geq Av - b \\
& \lambda \geq \mathbf{0}
\end{aligned} \tag{6}$$

The linear program (6) corresponds to the bilinear program (ABP–L_∞) with a fixed policy π and value function v.

Lemma 14. *Let $v \in \mathcal{K}$ be a transitive-feasible value function and let π be a policy. Then:*

$$f_1(\pi, v) \geq \|v - L_\pi v\|_\infty,$$

with an equality for a deterministic *policy π.*

Proof. The dual of the linear program (6) is the following program.

$$\begin{aligned}
\max_x \quad & x^\mathsf{T}(Av - b) \\
\text{s.t.} \quad & x \leq \pi \\
& \mathbf{1}^\mathsf{T} x = 1 \\
& x \geq \mathbf{0}
\end{aligned} \tag{7}$$

Note that replacing $\mathbf{1}^\mathsf{T} x = 1$ by $\mathbf{1}^\mathsf{T} x \leq 1$ preserves the properties of the linear program and would add an additional constraint in (ABP–L_∞): $\lambda' \geq 0$.

First, we show that $f_1(\pi, v) \geq \|L_\pi v - v\|_\infty$. Because v is feasible in the approximate bilinear program (ABP–L_∞), $Av - b \geq 0$ and $v \geq Lv$ from Lemma 38. Let state s be the state in which $t = \|L_\pi v - v\|_\infty$ is achieved. That is:

$$t = v(s) - \sum_{a \in \mathcal{A}} \pi(s, a) \left(r(s, a) + \sum_{s' \in \mathcal{S}} \gamma P(s', s, a) v(s') \right).$$

Now let $x(s, a) = \pi(s, a)$ for all $a \in \mathcal{A}$. This is a feasible solution with value t, from the stochasticity of the policy and therefore a lower bound on the objective value.

To show the equality for a deterministic policy π, we show that $f_1(\pi, v) \le \|Lv - v\|_\infty$, using that $\pi \in \{0, 1\}$. Then let x^* be an optimal solution of (7). Define the index of x^* with the largest objective value as:

$$i \in \arg \max_{\{i \,|\, x^*(i) > 0\}} (Av - b)(i).$$

Let solution $x'(i) = 1$ and $x'(j) = 0$ for $j \ne i$, which is feasible since $\pi(i) = 1$. In addition:

$$(Av - b)(i) = \|L_\pi v - v\|_\infty.$$

Now $(x^*)^\mathsf{T}(Av - b) \le (x')^\mathsf{T}(Av - b) = \|L_\pi v - v\|_\infty$, from the fact that i is the index of the largest element of the objective function. $\qquad \square$

When the policy π is fixed, the approximate bilinear program (ABP–L_∞) becomes the following linear program:

$$
\begin{aligned}
f_2(\pi) = \min_{\lambda, \lambda', v} \quad & \pi^\mathsf{T}\lambda + \lambda' \\
\text{s.t.} \quad & Av - b \ge \mathbf{0} \\
& \mathbf{1}\lambda + \lambda' \ge Av - b \\
& \lambda \ge \mathbf{0} \\
& v \in \mathcal{M}
\end{aligned}
\tag{8}
$$

Using Lemma 38 , this linear program corresponds to:

$$f_2(\pi) = \min_{v \in \mathcal{M} \cap \mathcal{K}} f_1(\pi, v).$$

Then it is easy to show that:

Lemma 15. *Given a policy π, let $\tilde{v}$ be an optimal solution of the linear program* (8). *Then:*

$$f_2(\pi) = \|L_\pi \tilde{v} - \tilde{v}\|_\infty \le \min_{v \in \mathcal{M} \cap \mathcal{K}} \|L_\pi v - v\|_\infty.$$

When v is fixed, the approximate bilinear program (ABP–L_∞) becomes the following linear program:

$$
\begin{aligned}
f_3(v) = \min_\pi \quad & f_2(\pi, v) \\
\text{s.t.} \quad & B\pi = \mathbf{1} \\
& \pi \ge \mathbf{0}
\end{aligned}
\tag{9}
$$

Note that the program is only meaningful if v is transitive-feasible and that the function f_2 corresponds to a minimization problem.

Lemma 16. *Let $v \in \mathcal{M} \cap \mathcal{K}$ be a transitive-feasible value function. There exists an optimal solution $\tilde{\pi}$ of the linear program* (9) *such that:*
1. *$\tilde{\pi}$ represents a* deterministic *policy*
2. *$L_{\tilde{\pi}} v = Lv$*
3. *$\|L_{\tilde{\pi}} v - v\|_\infty = \min_{\pi \in \Pi} \|L_\pi v - v\|_\infty = \|Lv - v\|_\infty$*

Proof. The existence of an optimal π that corresponds to a deterministic policy follows from Lemma 15, the correspondence between policies and values π, and the existence of a deterministic greedy policy.

Since $v \in \mathcal{K}$, we have for some policy π that:

$$v \geq Lv = L_\pi v \geq L_{\tilde{\pi}} v.$$

Assuming that $L_{\tilde{\pi}} < Lv$ leads to a contradiction since π is also a feasible solution in the linear program (9) and:

$$v - L_{\tilde{\pi}} > v - Lv$$
$$\|v - L_{\tilde{\pi}}\|_\infty > \|v - Lv\|_\infty.$$

This proves the lemma. $\qquad\square$

Theorem 13 now easily follows from the lemmas above.

Proof. Let $\bar{v}$ be a value function with the minimal $\|L\bar{v} - \bar{v}\|_\infty$ feasible in approximate bilinear program (ABP–L_∞), and let $\bar{\pi}$ be a greedy policy with respect to $\bar{v}$. Because $\bar{v} \geq v^*$, as Lemma 38 shows, we get:
$$t = \|L\bar{v} - \bar{v}\|_\infty = \|L_{\bar{v}}\bar{v} - \bar{v}\|_\infty.$$

Let f^* be the optimal objective value of (ABP–L_∞). Because both $\bar{v}$ and $\bar{\pi}$ are feasible in (ABP–L_∞), we have that $f^* \leq t$. Now, assume that $\tilde{v}$ is an optimal solution of (ABP–L_∞) with an objective value $\tilde{f} = \|L\tilde{v} - \tilde{v}\|_\infty > t$. Then, from Lemma 16, $\tilde{f} > t \geq f^*$, which contradicts the optimality of $\tilde{v}$.

To show that the optimal policy is deterministic and greedy, let π^* be the optimal policy. Then consider the state s for which $\tilde{\pi}$ does not define a deterministic greedy action. From the definition of greedy action $\bar{a}$:

$$(L_{\bar{a}}\tilde{v})(s) \leq (L_{\tilde{\pi}}\tilde{v})(s).$$

From the bilinear formulation (ABP–L_∞), it is easy to show that there is an optimal solution such that:

$$(L_a\tilde{v})(s) \leq \tilde{\lambda}' + \tilde{\lambda}(s, a)$$
$$\tilde{\lambda}(s, \bar{a}) \leq \tilde{\lambda}(s, a).$$

Then setting $\tilde{\pi}(s, \bar{a}) = 1$ and all other action probabilities to 0, the difference in the objective value function:
$$\tilde{\lambda}(s, \bar{a}) - \sum_{a \in \mathcal{A}} \tilde{\lambda}(s, a) \leq 0.$$

Therefore, the objective function for the deterministic greedy policy does not increase. The remainder of the theorem follows directly from Proposition 40, Proposition 41, and Proposition 42. The bounds on the policy loss then follow directly from Theorem 7. $\qquad\square$

4.2 Expected Policy Loss

This section describes bilinear programs that minimize 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.

$$
\begin{aligned}
\min_{\pi \,|\, \lambda, \lambda', v} \quad & \pi^\mathsf{T}\lambda + \lambda' - (1-\gamma)\alpha^\mathsf{T}v \\
\text{s.t.} \quad & B\pi = \mathbf{1} & Av - b \geq \mathbf{0} \\
& \pi \geq \mathbf{0} & \lambda + \lambda'\mathbf{1} \geq Av - b & \qquad \text{(ABP–}L_1\text{)} \\
& & \lambda, \lambda' \geq \mathbf{0} \\
& & v \in \mathcal{M}
\end{aligned}
$$

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

Theorem 17. *Given Assumption 10, any optimal solution $(\tilde{\pi}, \tilde{v}, \tilde{\lambda}, \tilde{\lambda}')$ of the approximate bilinear program* (ABP–L_1) *satisfies:*

$$
\frac{1}{1-\gamma}\left(\tilde{\pi}^\mathsf{T}\tilde{\lambda} + \tilde{\lambda}'\right) - \alpha^\mathsf{T}\tilde{v} = \frac{1}{1-\gamma}\|L\tilde{v} - \tilde{v}\|_\infty - \alpha^\mathsf{T}v \leq \min_{v \in \mathcal{K} \cap \mathcal{M}}\left(\frac{1}{1-\gamma}\|Lv - v\|_\infty - \alpha^\mathsf{T}v\right)
$$

$$
\leq \min_{v \in \mathcal{M}}\left(\frac{1}{1-\gamma}\|Lv - v\|_\infty - \alpha^\mathsf{T}v\right)
$$

Moreover, there exists an optimal solution $\tilde{\pi}$ that is greedy with respect to $\tilde{v}$ for which the policy loss is bounded by:

$$
\|v^* - v_{\tilde{\pi}}\|_{1,\alpha} \leq \frac{2}{1-\gamma}\left(\min_{v \in \mathcal{M}}\frac{1}{1-\gamma}\|Lv - v\|_\infty - \|v^* - v\|_{1,\alpha}\right).
$$

Notice that the bound in this theorem is tighter than the one in Theorem 13. In particular, $\|v^* - \tilde{v}\|_{1,\alpha} > 0$, unless the solution of the ABP is the optimal value function.

Proof. The proof of the theorem is almost identical to the proof of Theorem 13 with two main differences. First, the objective function of (ABP–L_1) is insensitive to adding a constant to the value function:

$$
\|(\tilde{v} + k\mathbf{1}) - L(v + k\mathbf{1})\|_\infty - \alpha^\mathsf{T}(v + k\mathbf{1}) = \|\tilde{v} - Lv\|_\infty - \alpha^\mathsf{T}v.
$$

Hence the missing factor 2 when going from minimization over $\mathcal{K} \cap \mathcal{M}$ to minimization over $\mathcal{M}$. The second difference is in the derivation of the bound on the policy loss, which follows directly from Theorem 8. $\qquad\square$

The bilinear program formulation in (ABP–L_1) can be further strengthened when an upper bound on the state-visitation frequencies is available.

$$
\begin{aligned}
\min_{\pi \,|\, \lambda, v} \quad & \pi^\mathsf{T} U \lambda - \alpha^\mathsf{T} v \\
\text{s.t.} \quad & B\pi = \mathbf{1} && Av - b \geq \mathbf{0} \\
& \pi \geq \mathbf{0} && \lambda \geq Av - b \\
& && \lambda \geq \mathbf{0} \\
& && v \in \mathcal{M}
\end{aligned}
\qquad \text{(ABP–}U\text{)}
$$

Here $U : |\mathcal{S}| \cdot |\mathcal{A}| \times |\mathcal{S}| \cdot |\mathcal{A}|$ is a matrix that maps a policy to bounds on state-action visitation frequencies. It must satisfy that:

$$
\pi(s, a) = 0 \Rightarrow (\pi^\mathsf{T} U)(s, a) = 0 \quad \forall s \in \mathcal{S} \; \forall a \in \mathcal{A}.
$$

Remark 18. One simple option is to have U represent a diagonal matrix of $\bar{u}$, where $\bar{u}$ is the bound for all policies $\pi \in \Pi$. That is:

$$
U((s, a), (s', a')) = \begin{cases} \bar{u}(s) & s' = s \\ 0 & \text{otherwise} \end{cases} \qquad \forall s, s' \in \mathcal{S} \; a, a' \in \mathcal{A}.
$$

The formal guarantees for this formulation are as follows.

Theorem 19. *Given Assumption 10 and that for all $\pi \in \Pi$: $\sum_{a \in \mathcal{A}} (\pi^\mathsf{T} U)(s, a) \geq u_\pi^\mathsf{T}(s)$, any optimal solution $(\tilde{\pi}, \tilde{v}, \tilde{\lambda}, \tilde{\lambda}')$ of the bilinear program (ABP–U) then satisfies:*

$$
\tilde{\pi}^\mathsf{T} U \tilde{\lambda} - \alpha^\mathsf{T} \tilde{v} = \| \tilde{v} - L\tilde{v} \|_{\bar{u}} - \alpha^\mathsf{T} \tilde{v} \leq \min_{v \in \mathcal{K} \cap \mathcal{M}} \left(\| v - Lv \|_{\bar{u}} - \alpha^\mathsf{T} v \right).
$$

Assuming that U is defined as in Remark 18, there exists an optimal solution $\tilde{\pi}$ that is greedy with respect to $\tilde{v}$ and:

$$
\| v^* - v_{\tilde{\pi}} \|_{1, \alpha} \leq \min_{v \in \mathcal{M} \cap \mathcal{K}} \left(\| v - Lv \|_{1, \bar{u}(v)} - \| v^* - v \|_{1, \alpha} \right).
$$

Here, $\bar{u}(v)$ represents an upper bound on the state-action visitation frequencies for a policy greedy with respect to value function v.

Unlike Theorem 13 and Theorem 17, the bounds in this theorem do not guarantee that the solution quality does not degrade by restricting the value function to be transitive-feasible.

To prove the theorem we first define the following linear program that solves for the L_1 norm of the Bellman update L_π for a value function v.

$$
\begin{aligned}
f_1(\pi, v) = \min_{\lambda, \lambda'} \quad & \pi^\mathsf{T} U \lambda \\
\text{s.t.} \quad & \mathbf{1}\lambda' + \lambda \geq Av - b \\
& \lambda \geq \mathbf{0}
\end{aligned}
\qquad \text{(10)}
$$

The linear program (6) corresponds to the bilinear program (ABP–U) with a fixed policy π and value function v. Notice, in particular, that $\alpha^\mathsf{T} v$ is a constant.

Lemma 20. *Let value function v be feasible in the bilinear program* (ABP–U), *and let π be an arbitrary policy. Then:*

$$f_1(\pi, v) \geq \|L_\pi v - v\|_{1,\bar{u}},$$

with an equality for a deterministic *policy.*

Proof. The dual of the linear program (10) program is the following.

$$
\begin{aligned}
\max_{x} \quad & x^\mathsf{T}(Av - b) \\
\text{s.t.} \quad & x \leq U^\mathsf{T}\pi \\
& x \geq \mathbf{0}
\end{aligned}
\tag{11}
$$

We have that $f_1(\pi, v) \geq \|L_\pi v - v\|_{1,\bar{u}}$ since $x = U^\mathsf{T}\pi$ is a feasible solution. To show the equality for a deterministic policy π, let x^* be an optimal solution of linear program (11). Since $Av \geq b$ and U is non-negative, an optimal solution satisfies $x = U^\mathsf{T}\pi$. The optimal value of the linear program thus corresponds to the definition of the weighted L_1 norm. $\square$

The proof of Theorem 19 is similar to the proof of Theorem 13, but using Theorem 8 instead of Theorem 7 to bound the policy loss. The existence of a deterministic and greedy optimal solution $\tilde{\pi}$ follows also like Theorem 13, omitting λ' and weighing λ by $\bar{u}$.

4.3 Hybrid Formulation

While the robust bilinear formulation (ABP–L_∞) guarantees to minimize the robust approximation error it may be overly pessimistic. The bilinear program (ABP–U), on the other hand, optimizes the average performance, but does not provide strong guarantees. It is possible to combine the advantages (and disadvantaged) of these programs using a hybrid formulation. The hybrid formulation minimizes the hybrid norm of the Bellman residual, defined as:

$$\|x\|_{k,c} = \max_{\{y \,|\, \mathbf{1}^\mathsf{T}y = k, \mathbf{1} \geq y \geq \mathbf{0}\}} \sum_{i=1}^{n} y(i)c(i)|x(i)|,$$

where n is the length of vector x and $c \geq \mathbf{0}$. It is easy to show that this norm represents the c-weighted L_1 norm of the k largest components of the vector. As such, it is more robust than the plain L_1 norm, but is not as sensitive to outliers as the L_∞ norm. Notice that the solution may be fractional when $k \notin \mathbb{Z}$ — that is, some elements are counted only partially.

The bilinear program that minimizes the hybrid norm is defined as follows.

$$
\begin{aligned}
\min_{\pi \,|\, \lambda, \lambda', v} \quad & \pi^\mathsf{T}U\lambda + k\lambda' \\
\text{s.t.} \quad & B\pi = \mathbf{1} & Av - b \geq \mathbf{0} \\
& \pi \geq \mathbf{0} & \lambda + \lambda'U^{-1}\mathbf{1} \geq Av - b \\
& & \lambda, \lambda' \geq \mathbf{0} \\
& & v \in \mathcal{M}
\end{aligned}
\tag{ABP–h}
$$

Here U is a matrix that maps a policy to bounds on state-action visitation frequencies, for example, as defined in Remark 18.

Theorem 21. *Given Assumption 10 and U that is defined as in Remark 18, any optimal solution $(\tilde{\pi}, \tilde{v}, \tilde{\lambda}, \tilde{\lambda}')$ of* (ABP–h) *then satisfies:*

$$\tilde{\pi}^\mathsf{T} U \tilde{\lambda} + k\tilde{\lambda}' = \|L\tilde{v} - \tilde{v}\|_{k,\bar{u}(\tilde{\pi})} \leq \min_{v \in \mathcal{M} \cap \mathcal{K}} \|L\tilde{v} - \tilde{v}\|_{k,\bar{u}(v)}.$$

Here, $\bar{u}(v)$ represents the upper bound on the state-action visitation frequencies for policy greedy with respect to value function v.

The implication of these bounds on the policy loss is beyond the scope of this paper, but it is likely that some form of policy loss bounds can be developed.

The proof of the theorem is almost identical to the proof of Theorem 13 lemma. We first define the following linear program, which solves for the required norm of the Bellman update L_π for value function v and policy π.

$$
\begin{aligned}
f_1(\pi, v) = \min_{\lambda, \lambda'} \quad & \pi^\mathsf{T} U \lambda + k\lambda' \\
\text{s.t.} \quad & \lambda' U^{-1} \mathbf{1} + \lambda \geq Av - b \\
& \lambda, \lambda' \geq \mathbf{0}
\end{aligned}
\tag{12}
$$

The linear program (12) corresponds to the bilinear program (ABP–h) with a fixed policy π and value function v.

Lemma 22. *Let $v \in \mathcal{K}$ be a transitive-feasible value function and let π be a policy and U be defined as in Remark 18. Then:*

$$f_1(\pi, v) \geq \|v - L_\pi v\|_{k,\bar{u}},$$

with an equality for a deterministic *policy π.*

Proof. The dual of the linear program (6) program is the following.

$$
\begin{aligned}
\max_{x} \quad & x^\mathsf{T}(Av - b) \\
\text{s.t.} \quad & x \leq U^\mathsf{T} \pi \\
& \mathbf{1}^\mathsf{T} \left(U^\mathsf{T}\right)^{-1} x \leq k \\
& x \geq \mathbf{0}
\end{aligned}
\tag{13}
$$

First, change the variables in the linear program to $x = U^\mathsf{T} z$ to get:

$$
\begin{aligned}
\max_{z} \quad & z^\mathsf{T} U(Av - b) \\
\text{s.t.} \quad & z \leq \pi \\
& \mathbf{1}^\mathsf{T} z \leq k \\
& z \geq \mathbf{0}
\end{aligned}
\tag{14}
$$

using the fact that U is diagonal and positive.

17

The norm$\|L_\pi v - v\|_{k,c}$ can be expressed as the following linear program:

$$
\begin{aligned}
\max_{y} \quad & y^\mathsf{T} X U(Av - b) \\
\text{s.t.} \quad & y \leq \mathbf{1} \\
& \mathbf{1}^\mathsf{T} y \leq k \\
& y \geq \mathbf{0}
\end{aligned}
\tag{15}
$$

Here, the matrix $X : |\mathcal{S}| \times |\mathcal{S}| \cdot |\mathcal{A}|$ selects the subsets of the Bellman residuals that correspond the the policy as defined:

$$
X(s, (s', a')) = \begin{cases} \pi(s', a') & \text{when } s = s' \\ 0 & \text{otherwise} \end{cases}.
$$

It is easy to shows that $v - L_\pi v = X(Av - b)$. Note that $XU = UX$ from the definition of U.

Clearly, when $\pi \in \{0, 1\}$ is deterministic the linear programs (14) and (15) are identical. When the policy π is stochastic, assume an optimal solution y of (15) and let $z = X^\mathsf{T} y$. Then, z is feasible in (14) with the identical objective value, which shows the inequality. $\quad\square$

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.

Solving a bilinear program is an NP-complete problem (Bennett and Mangasarian, 1992). 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 to 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 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 (BP–m), where $f(w, x, y, z)$ represents the objective function. The minimizations in the algorithm are linear

---

Algorithm 2: Iterative algorithm for solving (BP–m)

---

1 $(x_0, w_0) \leftarrow$ random ;

2 $(y_0, z_0) \leftarrow \arg\min_{y,z} f(w_0, x_0, y, z)$;

3 $i \leftarrow 1$;

4 **while** $y_{i-1} \neq y_i$ *or* $x_{i-1} \neq x_i$ **do**

5 $\quad\quad (y_i, z_i) \leftarrow \arg\min_{\{y,z \,|\, A_2 y + B_2 z = b_2 \; y,z \geq \mathbf{0}\}} f(w_{i-1}, x_{i-1}, y, z)$;

6 $\quad\quad (x_i, w_i) \leftarrow \arg\min_{\{x,w \,|\, A_1 x + B_1 w = b_1 \; x,w \geq \mathbf{0}\}} f(w, x, y_i, z_i)$;

7 $\quad\quad i \leftarrow i + 1$

8 **return** $f(w_i, x_i, y_i, z_i)$

---

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 23 (e.g. (Bennett and Mangasarian, 1992)). *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.

algorithm 2 can be further refined in case of approximate bilinear programs. For example, the constraint $v \in \mathcal{M}$ in the bilinear programs serves just to simplify the bilinear program and a value function that violates it may still be acceptable. The following proposition motivates the construction of a new value function from two transitive-feasible value functions.

Proposition 24. *Let $\tilde{v}_1$ and $\tilde{v}_2$ be feasible value functions in* (ABP–L_∞). *Then the value function*

$$\tilde{v}(s) = \min\{\tilde{v}_1(s), \tilde{v}_2(s)\}$$

is also feasible in bilinear program (ABP–L_∞). *Therefore $\tilde{v} \geq v^*$ and*

$$\|v^* - \tilde{v}\|_\infty \leq \min\left\{\|v^* - \tilde{v}_1\|_\infty, \|v^* - \tilde{v}_2\|_\infty\right\}.$$

The proof of the proposition is based on Jensen's inequality and is provided in the appendix. Note that $\tilde{v}$ may have a greater Bellman residual than either $\tilde{v}_1$ or $\tilde{v}_2$.

Proposition 24 can be used to extend algorithm 2 when solving ABPs. One option is to take the state-wise minimum of values from multiple random executions of algorithm 2, which preserves the transitive feasibility of the value function. However, the increasing number of value functions used to obtain $\tilde{v}$ also increases the potential sampling error.

6. Sampling Guarantees

In most practical problems, the number of states is too large to be explicitly enumerated. Even though the value function is restricted to be representable, the problem cannot be solved. The usual approach is to sample a limited number of states, actions, and their transitions to approximately calculate the value function. This section shows basic properties of the samples that can provide guarantees of the solution quality with incomplete samples.

First, we show a formal definition of the samples and then show how to use them. The simplest samples are defined as follows.

Definition 25. *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 $P(s, a)$.

More informative samples include the full distribution 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 26. *One-step samples with expectation* are defined as follows:

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

Membership a state in the samples is denoted simply as $s \in \Sigma$ 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.

Definition 27. The *sampled Bellman operator* and the corresponding set of sampled transitive-feasible functions are defined as:

$$(\bar{L}(v))(\bar{s}) = \max_{\{a \,|\, (\bar{s}, a) \in \bar{\Sigma}\}} r(\bar{s}, a) + \gamma \sum_{s' \in \mathcal{S}} P(\bar{s}, a, s') v(s') \qquad \forall \bar{s} \in \bar{\Sigma} \tag{16}$$

$$\bar{\mathcal{K}} = \left\{ v \,\middle|\, (\bar{s}, a, P(\bar{s}, a), r(\bar{s}, a)) \in \bar{\Sigma}, \ v(\bar{s}) \geq (\bar{L}v)(\bar{s}) \right\} \tag{17}$$

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

Definition 28. The *estimated Bellman operator* and the corresponding set of estimated transitive-feasible functions are defined as:

$$(\tilde{L}(v))(\bar{s}) = \max_{\{a \,|\, (\bar{s}, a) \in \tilde{\Sigma}\}} r(\bar{s}, a) + \gamma \frac{1}{n} \sum_{i=1}^{n} v(s_i) \qquad \forall \bar{s} \in \tilde{\Sigma} \tag{18}$$

$$\tilde{\mathcal{K}} = \left\{ v \,\middle|\, (\bar{s}, a, (s_1 \dots s_n), r(\bar{s}, a)) \in \tilde{\Sigma}, \ v(\bar{s}) \geq (\tilde{L}v)(\bar{s}) \right\} \tag{19}$$

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 assumed to be *undefined*.

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:

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

The 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 bounds on the number of violated constraints do not transform easily to the bounds on the quality of the value function, or the policy. In addition, the constraint distribution is often somewhat arbitrary because it is difficult to define and sampling from the appropriate distributions.

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 29 (Constraint Sampling Behavior)**.** For all representable value functions $v \in \mathcal{M}$:

$$\mathcal{K} \subseteq \bar{\mathcal{K}} \subseteq \mathcal{K}(\epsilon_p)$$

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

Assumption 30 (Constraint Estimation Behavior)**.** For all representable value functions $v \in \mathcal{M}$ the following holds:

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

These assumptions are intentionally made generic so that they apply to a wide range of scenarios. Domain specific assumptions are likely to lead to much tighter bounds, but these are beyond the scope of this paper.

Although we define the sampled Bellman operator directly, in practice only its approximate version is typically estimated. The direct definitions are defined only for the sake of theoretical analysis. The sampled matrices used in bilinear program (ABP–L_∞) are defined as follows for all $(s_i, a_j) \in \tilde{\Sigma}$.

$$\tilde{A}\Phi = \begin{pmatrix} - & \phi(s_i)^{\mathsf{T}} - \gamma \frac{1}{m} \sum_{s' \in s'_1 \dots s'_m} P(s_i, a_j, s') \phi(s')^{\mathsf{T}} & - \\ & \vdots & \end{pmatrix} \quad \tilde{b} = \begin{pmatrix} r(s_i, a_j) \\ \vdots \end{pmatrix}$$

$$\tilde{B}(s', (s_i, a_j)) = \mathrm{I}\left\{s' = s_i\right\} \quad \forall s' \in \tilde{\Sigma}$$

21

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

$$
\begin{aligned}
\min_{\pi \,|\, \lambda, \lambda', x} \quad & \pi^\mathsf{T}\lambda + \lambda' \\
\text{s.t.} \quad & \tilde{B}\pi = \mathbf{1} \qquad \tilde{A}\Phi x - b \geq \mathbf{0} \\
& \pi \geq \mathbf{0} \qquad \lambda + \lambda'\mathbf{1} \geq \tilde{A}\Phi x - \tilde{b} \\
& \lambda, \lambda' \geq \mathbf{0}
\end{aligned}
\qquad\qquad \text{(s–ABP–}L_\infty\text{)}
$$

The size of the bilinear program (s–ABP–L_∞) 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 (s–ABP–L_∞) is approximately three times the number of variables λ. Finally, the number of variables x corresponds to the number of approximation features.

Theorem 13 shows that sampled robust ABP minimizes $\|v - \tilde{L}v\|_\infty$ or $\|v - \bar{L}v\|_\infty$, depending on the samples used. It is then easy to derive sampling bounds that rely on the sampling assumptions defined above.

Theorem 31. *Let the optimal solutions to the sampled and precise Bellman residual minimization problems be:*

$$
v_1 \in \min_{v \in \mathcal{M} \cap \mathcal{K}} \|v - Lv\|_\infty \qquad v_2 \in \min_{v \in \mathcal{M} \cap \mathcal{K}} \|v - \bar{L}v\|_\infty \qquad v_3 \in \min_{v \in \mathcal{M} \cap \mathcal{K}} \|v - \tilde{L}v\|_\infty
$$

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 10, 29, and 30, the following holds:

$$
\|v^* - \hat{v}_1\|_\infty \leq \frac{2}{1-\gamma} \min_{v \in \mathcal{M}} \|v - Lv\|_\infty
$$

$$
\|v^* - \hat{v}_2\|_\infty \leq \frac{2}{1-\gamma} \left(\min_{v \in \mathcal{M}} \|v - Lv\|_\infty + \epsilon_p \right)
$$

$$
\|v^* - \hat{v}_3\|_\infty \leq \frac{2}{1-\gamma} \left(\min_{v \in \mathcal{M}} \|v - Lv\|_\infty + \epsilon_p + 2\epsilon_s \right)
$$

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 uniformity of state-samples.

Proof. We show bounds on $\|v_i - Lv_i\|_\infty$; the remainder of the theorem follows directly from Theorem 13. The second inequality follows from Assumption 29 and Lemma 36, as follows:

$$
\begin{aligned}
v_2 - Lv_2 &\leq v_2 - \bar{L}v_2 \\
&\leq v_1 - \bar{L}v_1 \\
&\leq v_1 - Lv_1 + \epsilon_p \mathbf{1}
\end{aligned}
$$

The second inequality follows from Assumptions 29, 30 and Lemma 36, as follows:

$$v_3 - Lv_3 \leq v_2 - \bar{L}v_2 + \epsilon_p \mathbf{1}$$
$$\leq v_2 - \tilde{L}v_2 + \epsilon_s \mathbf{1} + \epsilon_p \mathbf{1}$$
$$\leq v_1 - \tilde{L}v_1 + \epsilon_s \mathbf{1} + \epsilon_p \mathbf{1}$$
$$\leq v_1 - Lv_1 + 2\epsilon_s \mathbf{1} + \epsilon_p \mathbf{1}$$

Here, we use the fact that $v_i \geq Lv_i$ and that v_i's minimize the corresponding Bellman residuals. $\qquad\square$

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 Assumption 30 and Assumption 29. 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 issues with 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 (ALP–L_1) (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 drawback of using approximate bilinear programming, however, is the higher computational complexity.

The first and the second issue 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 (ABP–L_∞), on the other hand, has no such parameters. On the other hand, the expected-loss bilinear program (ABP–U) can be seen as a method for simultaneously optimizing c and the approximate linear program.

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 guar-

anteed to converge to a solution in finite time. As we show below, the exponential time complexity of ABP is unavoidable (unless P = NP).

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 32. *Assume $0 < \gamma < 1$, and a given $\epsilon > 0$. Then it is NP-complete to determine:*

$$\min_{v \in \mathcal{K} \cap \mathcal{M}} \|Lv - v\|_\infty < \epsilon \qquad \min_{v \in \mathcal{M}} \|Lv - v\|_\infty < \epsilon.$$

The problem remains NP-complete when Assumption 10 is satisfied. It is also NP-complete to determine:

$$\min_{v \in \mathcal{M}} \|Lv - v\|_\infty - \|v^* - v\|_{1,\alpha} < \epsilon \qquad \min_{v \in \mathcal{M}} \|Lv - v\|_{1,\bar{u}} - \|v^* - v\|_{1,\alpha} < \epsilon,$$

assuming that $\bar{u}$ is defined as in Remark 18.

As the theorem states, the value function approximation does not become computationally simpler even when Assumption 10 holds — a universal assumption in the paper. Notice that ALP can determine whether $\min_{v \in \mathcal{K} \cap \mathcal{M}} \|Lv - v\|_\infty = 0$ in polynomial time.

The proof of Theorem 32 is based on a reduction from SAT and can be found in Section A.2. The policy in the reduction determines the true literal in each clause, and the approximate value function corresponds to the truth value of the literals. The approximation basis forces literals that share the same variable to have consistent values.

Approximate bilinear programming can also improve on API with L_∞ minimization (L_∞-API for short), which is a leading 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):

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

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:

$$
\begin{aligned}
\min_{\phi, v} \quad & \phi \\
\text{s.t.} \quad & Av \geq b \quad (\equiv (\mathbf{I} - \gamma P_\pi)v \geq r_\pi \;\; \forall \pi \in \Pi) \\
& -(\mathbf{I} - \gamma P_\pi)v + \mathbf{1}\phi \geq -r_\pi \\
& v \in \mathcal{M}
\end{aligned}
\tag{20}
$$

In fact, OAPI corresponds to algorithm 2 applied to ABP because the linear program (20) corresponds to (ABP–L_∞) with a fixed π (see (8)). Then, using Proposition 23, we get the following corollary.

Corollary 33. *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 ϕ from above, and then it minimizes ϕ. L_∞-API constrains the Bellman residuals by ϕ 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.

L_∞-API cannot return an approximate value function that has a lower Bellman residual than ABP, given the optimality of ABP described in Theorem 13. However, even OAPI, an approximate ABP algorithm, performs comparably to L_∞-API, as the following theorem states.

Theorem 34. *Assume that L_∞-API converges to a policy π and a value function v that both satisfy: $\phi = \|v - L_\pi v\|_\infty = \|v - Lv\|_\infty$. Then*

$$\tilde{v} = v + \frac{\phi}{1 - \gamma}\mathbf{1}$$

is feasible in the bilinear program (ABP–L_∞), and it is a fixed point of OAPI. In addition, the greedy policies with respect to $\tilde{v}$ and v are identical.

Notice that while the optimistic and standard policy iterations can converge to the same solutions, the steps in their computation may not be identical. The actual results will depend on the initialization.

To prove the theorem, we first consider L_∞-API$_2$ as a modification of L_∞-API. L_∞-API$_2$ is shown in algorithm 1, where $\mathcal{Z}(\pi)$ is calculated using the following program:

$$
\begin{aligned}
\min_{\phi,v} \quad & \phi \\
\text{s.t.} \quad & (\mathbf{I} - \gamma P_a)v + \mathbf{1}\phi \geq r_a \quad \forall a \in \mathcal{A} \\
& -(\mathbf{I} - \gamma P_\pi)v + \mathbf{1}\phi \geq -r_\pi \\
& v \in \mathcal{M}
\end{aligned}
\tag{21}
$$

The difference between linear programs (4) and (27) is that (4) involves only the current policy, while (27) bounds $(\mathbf{I} - \gamma P_a)v + \mathbf{1}\phi \geq r_a$ from below for all policies. Linear program (27) differs from linear program (20) by not bounding the Bellman residual from below by $\mathbf{0}$.

Proposition 35. *L_∞-API and L_∞-API$_2$ generate the same sequence of policies if the initial policies and tie-breaking is the same.*

Proof. The proposition follows simply by induction from Lemma 39. The basic step follows directly from the assumption. For the inductive step, let $\pi_i^1 = \pi_i^2$, where π^1 and π^2 are the policies with (4) and (27). Then from Lemma 39, we have that the corresponding value functions $v_i^1 = v_i^2 + c\mathbf{1}$. Because π_{i+1}^1 and π_{i+1}^2 are chosen greedily, we have that $\pi_{i+1}^1 = \pi_{i+1}^2$. $\qquad\square$

The proof of Theorem 34 follows.

Proof. The proof is based on two facts. First, $\tilde{v}$ is feasible with respect to the constraints in (ABP–L_∞). The Bellman residual changes for all the policies identically, since a constant vector is added. Second, because L_π is greedy with respect to $\tilde{v}$, we have that $\tilde{v} \geq L_\pi \tilde{v} \geq L\tilde{v}$. The value function $\tilde{v}$ is therefore transitive-feasible.

From Proposition 45, L_∞-API can be replaced by L_∞-API$_2$, which will converge to the same policy π. L_∞-API$_2$ will converge to the value function

$$\tilde{v} = v + \frac{\phi}{1 - \gamma}\mathbf{1}.$$

From the constraints in (27) we have that $\tilde{v} \geq L_\pi \tilde{v}$. Then, since π is the greedy policy with regard to this value function, we have that $\tilde{v} \geq L_\pi \tilde{v} \geq L\tilde{v}$. Thus $\tilde{v}$ is transitive-feasible and feasible in (BP–m) according to Lemma 38. The theorem then follows from Lemma 39 and from the fact that the greedy policy minimizes the Bellman residual, as in the proof of Lemma 16. □

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 34 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. The focus of the paper is on the theoretical properties and the experiments are intentionally designed to avoid interaction between the approximation in the formulation and approximate solution methods. As Theorem 34 shows, even OAPI, the very simple approximate algorithm for ABP, can perform as well as existing methods on factored MDPs.

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

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 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 piecewise 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.
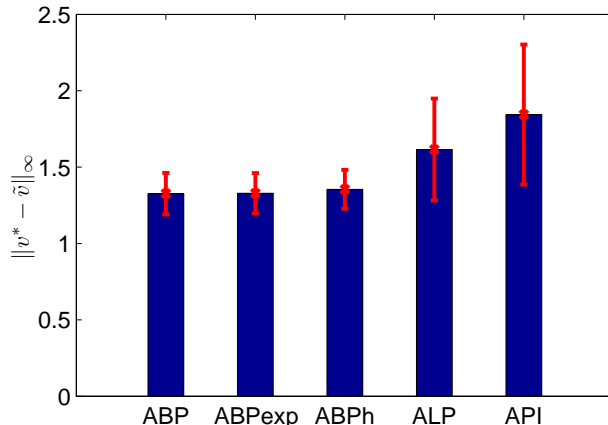
Figure 1: L_∞ Bellman residual for the chain problem

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 (ABP–L_1), and ABPhyb for (ABP–h) with $k = 5$. API denotes approximate policy iteration that minimizes the L_2 norm.

Figure 1 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 2 shows the expected policy loss, as specified in Definition 5, 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 3 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 ABP formulations, even though their Bellman residual is significantly higher. This is possible since ABP simply minimizes bounds on the policy loss. The analysis of tightness of policy loss bounds is beyond the scope of this paper.

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 $\gamma = 0.99$.

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 the number of the approximation features 100 and 144, which corresponds 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
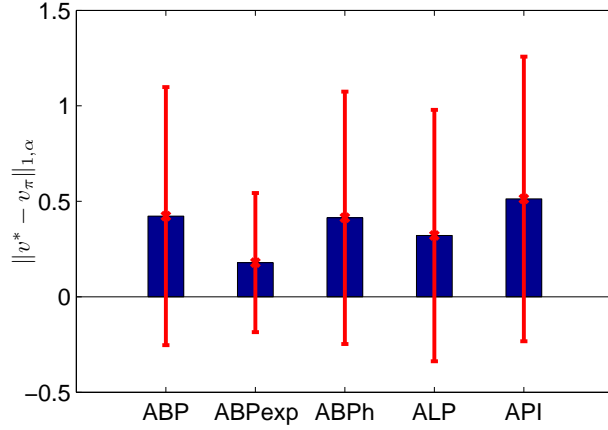
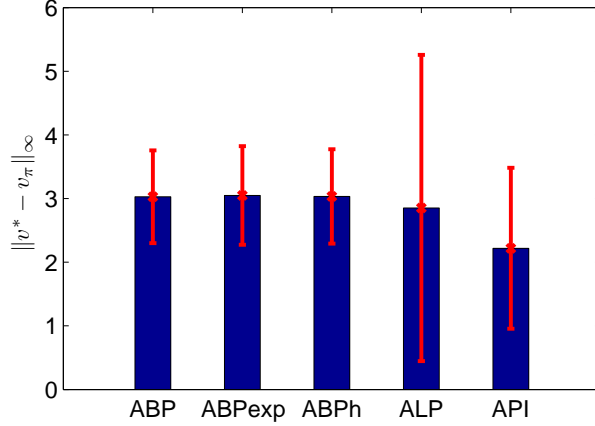Figure 2: Expected policy loss for the chain problem

Figure 3: Robust policy loss for the chain problem

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

9. Conclusion and Future Work

We proposed and analyzed 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 showed that there is no asymptotically simpler formulation, since finding the closest value function and

(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.

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 discussed approximate solutions of the ABP, a deeper study of bilinear solvers may render optimal solution methods feasible. ABPs have a small number of essential variables (that determine the value function) and a large number of constraints, which can be leveraged by the solvers (Petrik and Zilberstein, 2007). 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.

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

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 Olvi L. Mangasarian. Bilinear separation of two sets in n-space. Technical report, Computer Science Department, University of Wisconsin, 1992.

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 Pucci 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 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.

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.

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.

Appendix A. Proofs

A.1 Properties of Transitive-Feasible Value Functions

Basic properties of the Bellman operator, which we often use without a reference are the following.

Lemma 36. *Let v be any value function and let c be a scalar. Then:*

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

Lemma 37 (Monotonicity). *Let P be a stochastic matrix. Then both the linear operators P and $(I - \gamma P)^{-1}$ are monotonous:*

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

for all x and y.

Lemma 4. *Transitive feasible value functions form an upper bound on the optimal value function. If $v \in \mathcal{K}(\epsilon)$ is an ϵ-transitive-feasible value function, then*

$$v \geq v^* - \frac{\epsilon}{1 - \gamma}\mathbf{1}.$$

Proof. Let P^* and r^* be the transition matrix and the reward vector of the policy. Then, we have using Lemma 37:

$$v \geq Lv - \epsilon\mathbf{1}$$
$$v \geq \gamma P^* v + r^* - \epsilon\mathbf{1}$$
$$(\mathbf{I} - \gamma P^*)v \geq r^* - \epsilon\mathbf{1}$$
$$v \geq (\mathbf{I} - \gamma P^*)^{-1} r^* - \frac{\epsilon}{1 - \gamma}$$

$\square$

Lemma 38. *A value function v satisfies $Av \geq b$ if an only if $v \geq Lv$. In addition, if v is feasible in* (ABP–L_∞)*, then $v \geq v^*$.*

Proof. The backward implication of the first part of the lemma follows directly from the definition. The forward implication follows by an existence of $\lambda = \mathbf{0}$, $\lambda' = \| [Av - r]_+ \|_\infty$, which satisfy the constraints. The constraints on π are independent and therefore can be satisfied independently. The second part of the lemma also holds in ALPs (de Farias, 2002) and is proven identically. $\square$

The minimization $\min_{v \in \mathcal{M}} \|Lv - v\|_\infty$ for a policy π can be represented as the following linear program.

$$
\begin{aligned}
\min_{\phi,v} \quad & \phi \\
\text{s.t.} \quad & (\mathbf{I} - \gamma P_\pi)v + \mathbf{1}\phi \geq r_\pi \\
& -(\mathbf{I} - \gamma P_\pi)v + \mathbf{1}\phi \geq -r_\pi \\
& v \in \mathcal{M}
\end{aligned}
\tag{22}
$$

Consider also the following linear program.

$$
\begin{aligned}
\min_{\phi,v} \quad & \phi \\
\text{s.t.} \quad & (\mathbf{I} - \gamma P_\pi)v \geq r_\pi \\
& -(\mathbf{I} - \gamma P_\pi)v + \mathbf{1}\phi \geq -r_\pi \\
& v \in \mathcal{M}
\end{aligned}
\tag{23}
$$

Next we show that the optimal solutions of (22) and (23) are closely related.

Lemma 39. *Assume Assumption 10 and a given policy π. Let ϕ_1, v_1 and ϕ_2, v_2 optimal solutions of linear programs (22) and (23) respectively. Define:*

$$
\bar{v}_1 = v_1 + \frac{\phi_1}{1 - \gamma} \qquad\qquad \bar{v}_2 = v_1 - \frac{\phi_2}{2(1 - \gamma)}\mathbf{1}
$$

Then:
1. *$2\phi_1 = \phi_2$*
2. *$\bar{v}_1$ is an optimal solution in (23).*
3. *$\bar{v}_2$ is an optimal solution in (22).*
4. *Greedy policies with respect to v_1 and $\bar{v}_1$ are identical.*
5. *Greedy policies with respect to v_2 and $\bar{v}_2$ are identical.*

Proof. Let $\bar{\phi}_1 = 2\phi_1$ and $\bar{\phi}_2 = \frac{\phi_2}{2}$. We first show $\bar{\phi}_1, \bar{v}_1$ is feasible in (23). It is representable since $\mathbf{1} \in \mathcal{M}$ and it is feasible by the following simple algebraic manipulation:

$$
\begin{aligned}
(\mathbf{I} - \gamma P_\pi)\bar{v}_1 &= (\mathbf{I} - \gamma P_\pi)v_1 + (\mathbf{I} - \gamma P_\pi)\frac{\phi_1}{1 - \gamma}\mathbf{1} \\
&= (\mathbf{I} - \gamma P_\pi)v_1 + \phi_1\mathbf{1} \\
&\geq -\phi_1\mathbf{1} + r_\pi + \phi_1\mathbf{1} \\
&= r_\pi
\end{aligned}
$$

and

$$
\begin{aligned}
-(\mathbf{I} - \gamma P_\pi)\bar{v}_1 + \bar{\phi}_1\mathbf{1} &= -(\mathbf{I} - \gamma P_\pi)\bar{v}_1 + 2\phi_1\mathbf{1} \\
&= -(\mathbf{I} - \gamma P_\pi)v_1 - (\mathbf{I} - \gamma P_\pi)\frac{\phi_1}{1 - \gamma}\mathbf{1} + 2\phi_1\mathbf{1} \\
&= -(\mathbf{I} - \gamma P_\pi)v_1 - \phi_1\mathbf{1} + 2\phi_1\mathbf{1} \\
&\geq -\phi_1\mathbf{1} - r_\pi + 2\phi_1\mathbf{1} \\
&= -r_\pi
\end{aligned}
$$

33

Next we show that $\bar{\phi}_2, \bar{v}_2$ is feasible in (22). This solution is representable, since $\mathbf{1} \in \mathcal{M}$, and it is feasible by the following simple algebraic manipulation:

$$
\begin{aligned}
(\mathbf{I} - \gamma P_\pi)\bar{v}_2 + \bar{\phi}_2 \mathbf{1} &= (\mathbf{I} - \gamma P_\pi)v_2 - (\mathbf{I} - \gamma P_\pi)\frac{\phi_2}{2(1-\gamma)}\mathbf{1} + \frac{\phi_2}{2}\mathbf{1} \\
&= (\mathbf{I} - \gamma P_\pi)v_2 - \frac{\phi_2}{2}\mathbf{1} + \frac{1}{2}\bar{\phi}_2 \mathbf{1} \\
&= (\mathbf{I} - \gamma P_\pi)v_2 \\
&\geq r_\pi
\end{aligned}
$$

and

$$
\begin{aligned}
-(\mathbf{I} - \gamma P_\pi)\bar{v}_2 + \bar{\phi}_2 \mathbf{1} &= -(\mathbf{I} - \gamma P_\pi)\bar{v}_2 + \frac{\phi_2}{2}\mathbf{1} \\
&= -(\mathbf{I} - \gamma P_\pi)v_2 - (\mathbf{I} - \gamma P_\pi)\frac{\phi_2}{1-\gamma}\mathbf{1} + \frac{\phi_2}{2}\mathbf{1} \\
&= -(\mathbf{I} - \gamma P_\pi)v_2 + \phi_2 \mathbf{1} \\
&\geq -r_\pi
\end{aligned}
$$

It is now easy to shows that $\bar{\phi}_1, \bar{v}_1$ is optimal by contradiction. Assume that there exists a solution $\phi_2 < \bar{\phi}_1$. But then:

$$
2\bar{\phi}_2 \leq \phi_2 < \bar{\phi}_1 \leq 2\phi_1,
$$

which is a contradiction with the optimality of ϕ_1. The optimality of $\bar{\phi}_2, \bar{v}_2$ can be shown similarly. $\qquad\square$

Proposition 40. *Assumption 10 implies that:*

$$
\min_{v \in \mathcal{M} \cap \mathcal{K}} \|Lv - v\|_\infty \leq 2 \min_{v \in \mathcal{M}} \|Lv - v\|_\infty.
$$

Proof. Let $\hat{v}$ be the minimizer of $\hat{\phi} = \min_{v \in \mathcal{M}} \|Lv - v\|_\infty$, and let $\hat{\pi}$ be a policy that is greedy with respect to $\hat{v}$. Define:

$$
\tilde{v} = \hat{v} + \frac{\hat{\phi}}{1 - \gamma}.
$$

Then from Lemma 39:
   1. Value function $\tilde{v}$ is an optimal solution of (23): $\tilde{v} \geq L_\pi \tilde{v}$
   2. Policy $\hat{\pi}$ is greedy with regard to $\tilde{v}$: $L_{\hat{\pi}} \tilde{v} \geq L \tilde{v}$
   3. $\|L_\pi \tilde{v} - \tilde{v}\|_\infty = 2\hat{\phi}$
Then using a simple algebraic manipulation:

$$
\tilde{v} \geq L_{\hat{\pi}} \tilde{v} = L \tilde{v}
$$

and the proposition follows from Lemma 38. $\qquad\square$

Proposition 41. *Let $\tilde{v}$ be a solution of the approximate bilinear program* (ABP–L_∞) *and let:*

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

Then:

1. $\|Lv' - v'\|_\infty = \frac{\|Lv-v\|_\infty}{2}$.
2. *Greedy policies with respect to v and v' are identical.*

The proposition follows directly from Lemma 39.

Proposition 42. *Assumption 10 implies that:*

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

Proof. Assume that $\hat{v}$ is the minimizer of $\min_{v\in\mathcal{M}} \|v - v^*\|_\infty \leq \epsilon$. Then:

$$
\begin{aligned}
v^* - \epsilon\mathbf{1} &\leq & v & \leq v^* + \epsilon\mathbf{1}\\
Lv^* - \gamma\epsilon\mathbf{1} &\leq & Lv & \leq Lv^* + \gamma\epsilon\mathbf{1}\\
Lv^* - \gamma\epsilon\mathbf{1} - v &\leq & Lv - v & \leq Lv^* + \gamma\epsilon\mathbf{1} - v\\
Lv^* - v^* - (1+\gamma)\epsilon\mathbf{1} &\leq & Lv - v & \leq Lv^* - v^* + (1+\gamma)\epsilon\mathbf{1}\\
-(1+\gamma)\epsilon\mathbf{1} &\leq & Lv - v & \leq (1+\gamma)\epsilon\mathbf{1}.
\end{aligned}
$$

$\square$

Proposition 24. *Let $\tilde{v}_1$ and $\tilde{v}_2$ be feasible value functions in* (ABP–L_∞). *Then the value function*

$$\tilde{v}(s) = \min\{\tilde{v}_1(s), \tilde{v}_2(s)\}$$

is also feasible in bilinear program (ABP–L_∞). *Therefore $\tilde{v} \geq v^*$ and*

$$\|v^* - \tilde{v}\|_\infty \leq \min\left\{\|v^* - \tilde{v}_1\|_\infty, \|v^* - \tilde{v}_2\|_\infty\right\}.$$

Proof. Consider a state s and action a. Then from transitive feasibility of the value functions $\tilde{v}_1$ and $\tilde{v}_2$ we have:

$$\tilde{v}_1(s) \geq \gamma \sum_{s'\in\mathcal{S}} P(s', a, a)\tilde{v}_1(s') + r(s, a)$$

$$\tilde{v}_2(s) \geq \gamma \sum_{s'\in\mathcal{S}} P(s', a, a)\tilde{v}_2(s') + r(s, a).$$

From the convexity of the min operator we have that:

$$\min\left\{\sum_{s'\in\mathcal{S}} P(s', a, a)\tilde{v}_1, \sum_{s'\in\mathcal{S}} P(s', a, a)\tilde{v}_2(s')\right\} \geq \sum_{s'\in\mathcal{S}} P(s', a, a)\min\{\tilde{v}_1(s'), \tilde{v}_2)(s')\}.$$

Then the proposition follows by the following simple algebraic manipulation:

$$
\begin{aligned}
\tilde{v} = \min\{\tilde{v}_1(s), \tilde{v}_2(s)\} &\geq \gamma\min\left\{\sum_{s'\in\mathcal{S}} P(s', a, a)\tilde{v}_1, \sum_{s'\in\mathcal{S}} P(s', a, a)\tilde{v}_2(s')\right\} + r(s, a)\\
&\geq \gamma \sum_{s'\in\mathcal{S}} P(s', a, a)\min\{\tilde{v}_1(s'), \tilde{v}_2)(s')\} + r(s, a)\\
&= \gamma \sum_{s'\in\mathcal{S}} P(s', a, a)\tilde{v}(s) + r(s, a).
\end{aligned}
$$

$\square$

Lemma 43. *Let u_π be the state-action visitation frequency of policy π. Then:*

$$\mathbf{1}^\mathsf{T} u = \frac{1}{1-\gamma}.$$

Proof. Let $u_a(s) = u_\pi(s, \pi(s, a))$ for all states $s \in \mathcal{S}$ and actions $a \in \mathcal{A}$. The lemma follows as:

$$\sum_{a \in \mathcal{A}} u_a^\mathsf{T}(\mathbf{I} - \gamma P_a) = c^\mathsf{T}$$

$$\sum_{a \in \mathcal{A}} u_a^\mathsf{T}(\mathbf{I} - \gamma P_a)\mathbf{1} = c^\mathsf{T}\mathbf{1}$$

$$(1 - \gamma) \sum_{a \in \mathcal{A}} u_a^\mathsf{T}\mathbf{1} = 1$$

$$u^\mathsf{T}\mathbf{1} = \frac{1}{1-\gamma}.$$

$\square$

A.2 NP-Completeness

Proposition 44 (e.g. (Mangasarian, 1995)). *A bilinear program can be solved in NP time.*

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.

Theorem 32. *Assume $0 < \gamma < 1$, and a given $\epsilon > 0$. Then it is NP-complete to determine:*

$$\min_{v \in \mathcal{K} \cap \mathcal{M}} \|Lv - v\|_\infty < \epsilon \qquad \min_{v \in \mathcal{M}} \|Lv - v\|_\infty < \epsilon.$$

The problem remains NP-complete when Assumption 10 is satisfied. It is also NP-complete to determine:

$$\min_{v \in \mathcal{M}} \|Lv - v\|_\infty - \|v^* - v\|_{1,\alpha} < \epsilon \qquad \min_{v \in \mathcal{M}} \|Lv - v\|_{1,\bar{u}} - \|v^* - v\|_{1,\alpha} < \epsilon,$$

assuming that $\bar{u}$ is defined as in Remark 18.

Proof. The membership in NP follows from Theorem 13 and Proposition 44. We show NP-hardness by a reduction from the 3SAT problem. We first don't assume Assumption 10. We show the theorem for $\epsilon = 1$. 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 is satisfiable. The value of the states will correspond to the truth value of the literals and clauses. The approximation features ϕ will be used to constraint the values of literals that share the same variable. The MDP constructed from the SAT formula is depicted in Figure 4.

$$(\ l_{11} \ \lor \ l_{12} \ \lor \ l_{13}) \ \land \ (l_{21} \ \lor \ l_{22} \ \lor \ l_{23})$$
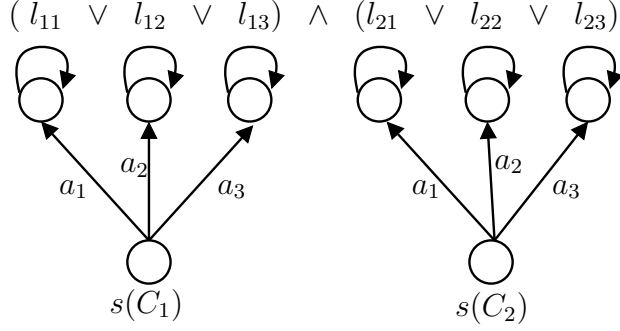
Figure 4: MDP constructed from the corresponding SAT formula.

Consider a SAT problem with clauses C_i:

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

where l_{ij} are literals. A literal is a variable or a negation of a variable. The variables in the SAT are $x_1 \dots x_m$. 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:

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

There are 3 actions available for each state $s(C_i)$, which determine the true literal of the clause. There is only a single action available in states $s(l_{ij})$ and $\bar{s}$. All transitions in the MDP 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 the following:

$$t(s(C_i), a_j) = (s(l_{ij}), 1 - \gamma) \tag{24}$$

$$t(s(l_{ij}), a) = (s(l_{ij}), -(1 - \gamma)) \tag{25}$$

$$t(\bar{s}, a) = (\bar{s}, 2 - \gamma) \tag{26}$$

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:

$$\phi_k(s(C_i)) = 0$$

$$\phi_k(\bar{s}) = 0$$

$$\phi_k(s(l_{ij})) = \begin{cases} 1 & \text{if } l_{ij} = x_k \\ -1 & \text{if } l_{ij} = \neg x_k \end{cases}$$

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

$$\bar{\phi}(s(C_i)) = 1$$

$$\bar{\phi}(s(l_{ij})) = 0$$

$$\bar{\phi}(\bar{s}) = 1.$$

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 $\min_{v \in \mathcal{M} \cap \mathcal{K}} \|Lv - v\|_\infty < 1$. 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:

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

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

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

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 $\min_{v \in \mathcal{K} \cap \mathcal{M}} \|Lv - v\|_\infty \geq 1 - \frac{\gamma^2}{2}$. Now, given a value function v, there are two possible cases for each $v(s(l_{ij}))$: 1) a positive value, 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 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, $\min_{v \in \mathcal{M} \cap \mathcal{K}} \|Lv - v\|_\infty \leq 1 - \gamma^2$ is and only if the 3-SAT problem is feasible.

It remains to show that the problem remains NP-complete even when Assumption 10 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 $\min_{v \in \mathcal{M} \cap \mathcal{K}} \|Lv - v\|_\infty \leq 1 - \gamma^2$ 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:

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

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 $s(C_i) \rightarrow s(l_{ij})$ 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 is unsatisfiable, the Bellman residual is at least $1 - \frac{\gamma^2}{2}$.

The NP-completeness of $\min_{v \in \mathcal{M}} \|Lv - v\|_\infty < \epsilon$ follows trivially from Proposition 40. 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 $v \in \mathcal{M}$. 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 visitation 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 $\bar{u}(l_{ij} > 0$. $\qquad\square$

A.3 Equivalence of OAPI and API

We first consider L_∞-API$_2$ as a modification of L_∞-API. L_∞-API$_2$ is shown in algorithm 1, where $f(\pi)$ is calculated using the following program:

$$
\begin{aligned}
\min_{\phi,v} \quad & \phi \\
\text{s.t.} \quad & (\mathbf{I} - \gamma P_\pi)v + \mathbf{1}\phi \geq r_\pi \\
& -(\mathbf{I} - \gamma P_\pi)v + \mathbf{1}\phi \geq -r_\pi \\
& v \in \mathcal{M}
\end{aligned} \tag{27}
$$

Proposition 45. *L_∞-API and L_∞-API$_2$ generate the same sequence of policies if the initial policies and tie-breaking is the same.*

Proof. The proposition follows simply by induction from Lemma 39. The basic step follows directly from the assumption. For the inductive step, let $\pi_i^1 = \pi_i^2$, where π^1 and π^2 are the policies with (4) and (27). Then from Lemma 39, we have that the corresponding value functions $v_i^1 = v_i^2 + c\mathbf{1}$. Because π_{i+1}^1 and π_{i+1}^2 are chosen greedily, we have that $\pi_{i+1}^1 = \pi_{i+1}^2$. $\qquad\square$

We are ready now to prove the theorem.

Theorem 34. *Assume that L_∞-API converges to a policy π and a value function v that both satisfy: $\phi = \|v - L_\pi v\|_\infty = \|v - Lv\|_\infty$. Then*

$$
\tilde{v} = v + \frac{\phi}{1 - \gamma}\mathbf{1}
$$

is feasible in the bilinear program (ABP–L_∞), and it is a fixed point of OAPI. In addition, the greedy policies with respect to $\tilde{v}$ and v are identical.

Proof. From Proposition 45, L_∞-API can be replaced by L_∞-API$_2$, which will converge to the same policy π. L_∞-API$_2$ will converge to the value function

$$
\tilde{v} = v + \frac{\phi}{1 - \gamma}\mathbf{1}.
$$

From the constraints in (27) we have that:

$$\tilde{v} \geq L_\pi \tilde{v}.$$

Then, since π is the greedy policy with regard to this value function, we have that:

$$\tilde{v} \geq L_\pi \tilde{v} \geq L\tilde{v}.$$

Thus $\tilde{v}$ is transitive-feasible and feasible in (BP–m) according to Lemma 38. The theorem then follows from Lemma 39 and from the fact that the greedy policy minimizes the Bellman residual, as in the proof of Lemma 16. $\square$

Notice that while the optimistic and standard policy iterations can converge to the same solutions, the steps in their computation may not be identical. The actual results will depend on the initialization.