Technical Report DCS-TR-631

Prioritized Sweeping Converges to the Optimal Value Function

Lihong Li and **Michael L. Littman**

{lihong,mlittman}@cs.rutgers.edu
RL^3 Laboratory
Department of Computer Science
Rutgers University
Piscataway, NJ 08854

First created: April 30, 2008
Last modified: June 15, 2008

Abstract

Prioritized sweeping (PS) and its variants are model-based reinforcement-learning algorithms that have demonstrated superior performance in terms of computational and experience efficiency in practice. This note establishes the first—to the best of our knowledge—formal proof of convergence to the optimal value function when they are used as *planning* algorithms. We also describe applications of this result to provably efficient model-based reinforcement learning in the PAC-MDP framework. We do not address the issue of convergence rate in the present paper.

Keywords: prioritized sweeping, asynchronous dynamic programming, asymptotic convergence, decision-theoretic planning, Markov decision process.

Prioritized Sweeping Converges to the Optimal Value Function

Lihong Li and Michael L. Littman

1 Introduction

Prioritized sweeping is a model-based reinforcement-learning algorithm, first proposed by Moore and Atkeson (1993), and has demonstrated superior performance in practice. At a high level, this algorithm attempts to make the best use of its limited computational power to approximate the optimal value function of a Markov decision process by adjusting state values whose Bellman errors are large. This heuristic is believed to be the key to its efficiency and has been used in a similar way by related algorithms such as Queue-Dyna (Peng & Williams, 1993), generalized prioritized sweeping (GenPS) (Andre, Friedman, & Parr, 1998), and general prioritized solvers (Wingate & Seppi, 2005). We call these algorithms *prioritized-sweeping algorithms*. We refer to the algorithm of Moore and Atkeson (1993) as PS.

Despite their empirical successes, it remains an open problem whether these algorithms converge to the optimal value function in the limit. Although it is often believed that such a proof follows from known results in asynchronous dynamic programming (ADP) (Bertsekas & Tsitsiklis, 1989), as we will argue later, this connection is not obvious. The key obstacle is that prioritized-sweeping algorithms often do not guarantee that every state value is updated infinitely often, which is usually required for convergence of ADP algorithms.

Convergence of prioritized-sweeping algorithms can be trivially guaranteed by interleaving prioritized updates with uniform updates[1]. However, allowing uniform updates would seem to degrade performance in practice since the efficiency of these algorithms largely comes from the informative ordering of updates they adopt. Establishing convergence for "pure" prioritized-sweeping algorithms, therefore, eliminates the need for such a somewhat artificial fix and simplifies implementation of these algorithms.

Although prioritized-sweeping algorithms were originally proposed for reinforcement learning, their convergence property is better understood in the planning setting where a complete world model is given as input. We will focus on planning in most part of the paper, but will also mention how we may apply these convergence results in planning to obtain provably efficient reinforcement-learning algorithms.

The rest of the paper is organized as follows. Section 2 gives the notation we use, describes a generic prioritized-sweeping algorithms for planning, and argues why convergence of these algorithms is not as direct as generally believed. Section 3 provides a general convergence result for a family of prioritized sweeping algorithms, and then applies it to establish convergence of PS and GenPS. Finally, we briefly

[1] C. Szepesvári, personal communication, 2008.

show how these convergence results allow us to obtain efficient reinforcement-learning algorithms in Section 4.

2 Preliminaries

A Markov decision process (Puterman, 1994), or MDP for short, is described as a five tuple: $M = \langle S, A, T, R, \gamma \rangle$, where S is a finite set of states; A is a finite set of actions; T is the transition function with $T(\cdot|s,a)$ denoting the next-state distribution after taking action a in state s; R is a reward function with $R(s,a)$ denoting the expected immediate reward gained by taking action a in state s; and $\gamma \in [0,1)$ is a discount factor. Without loss of generality, we assume $0 \leq R(s,a) \leq 1$ for all state–action pairs (s,a).

In this and the next section, we will assume the world model, namely the five-tuple in M, is known to the algorithms. The goal in this setting is to compute the optimal value function, $V^* : S \to \mathbb{R}$, that is the unique fixed point of the *Bellman equation* (Bellman, 1957):

$$\forall s \in S: \quad V^*(s) = \max_{a \in A} \left[R(s,a) + \gamma \sum_{s' \in S} T(s'|s,a) V^*(s') \right].$$

A value function $V \neq V^*$ does not satisfy the Bellman equation and incurs a *Bellman error* in state s, which is defined by

$$E(s;V) = \max_{a \in A} \left[R(s,a) + \gamma \sum_{s' \in S} T(s'|s,a) V(s') \right] - V(s).$$

Asynchronous value iteration, sometimes known as *Gauss-Seidel value iteration* (Puterman, 1994), is able to compute V^*. Starting from an arbitrary initial value function V, it iteratively sweeps over the entire state space to perform *Bellman backups* in every state:

$$\forall s \in S: \quad V(s) \leftarrow \max_{a \in A} \left[R(s,a) + \gamma \sum_{s' \in S} T(s'|s,a) V(s') \right].$$

Although the function V computed by this algorithm converges to V^* in the limit, it is impractical when $|S|$ is large. A powerful technique is to perform Bellman backups in *more useful* states, in the hope of accelerating convergence with the same amount of computational resources. This idea motivates prioritized sweeping, described next.

2.1 Prioritized-Sweeping Algorithms

Algorithm 1 gives a generic prioritized-sweeping algorithm that unifies a number of closely related algorithms. This algorithm requires a nonnegative, real-valued priority function, $H : S \to \mathbb{R}_+$, which is used to order states for Bellman backups. In general, a larger $H(s)$ implies higher priority to perform a Bellman backup on state s. Different definitions and/or update rules of $H(s)$ define different algorithms, and a good priority function can result in faster convergence of V_t to V^*, in practice. We first illustrate how specific algorithms can be derived by instantiating H in Algorithm 1.

Algorithm 1 An Abstract Prioritized-Sweeping Algorithm.

Inputs: $M = \langle S, A, T, R, \gamma \rangle$
Initialize value function $V_1(s)$ for all states $s \in S$.
Initialize priority values $H_1(s)$ for all states $s \in S$.
for $t = 1, 2, 3, \cdots$ **do**
  Pick the state with the highest priority: $s_t \leftarrow \arg\max_{s \in S} H_t(s)$.
  Perform Bellman backup on state s_t: $V_{t+1}(s_t) \leftarrow \max_{a \in A} \left[R(s_t, a) + \gamma \sum_{s' \in S} T(s'|s_t, a) V_t(s') \right]$.
  For states s other than s_t: $V_{t+1}(s) \leftarrow V_t(s)$.
  Update priority values for all states $s \in S$, and denote the new values $H_{t+1}(s)$.
end for

1. Asynchronous value iteration described in the previous subsection repeatedly performs Bellman backups in a *fixed* state ordering, which is easily guaranteed by many $H(s)$ functions. For example, let $S = \{1, 2, \cdots, n\}$ and suppose we wish to perform Bellman backups in the order of

$$1 \to 2 \to \cdots \to n-1 \to n \to 1 \to 2 \to \cdots,$$

   then this is guaranteed by the following priority function: it is initialized by:

$$\forall s \in S: \quad H_1(s) \leftarrow \frac{2n - s}{2n} \tag{1}$$

   and updated by:

$$\forall s \in S: \quad H_{t+1}(s) \leftarrow \begin{cases} H_t(s), & \text{if } s \neq s_t \\ \frac{H_t(s)}{2} & \text{if } s = s_t \end{cases}. \tag{2}$$

   Therefore, asynchronous value iteration can be viewed as a special case of Algorithm 1.

2. PS does not specify how to initialize H. In practice, we may initialize $H(s)$ with random positive values.[2] The algorithm then updates $H(s)$ in the following way:

$$\forall s \in S: \quad H_{t+1}(s) \leftarrow \begin{cases} \max\left\{H_t(s), \Delta_t \cdot \max_{a \in A} T(s_t|s, a)\right\}, & \text{if } s \neq s_t \\ \Delta_t \cdot \max_{a \in A} T(s_t|s, a), & \text{if } s = s_t \end{cases}. \tag{3}$$

   where $\Delta_t = |V_{t+1}(s_t) - V_t(s_t)| = |E(s_t; V_t)|$ is the change of s_t's value after the most recent Bellman backup.

3. GenPS updates $H(s)$ so that it is always the absolute Bellman error in state s:

$$\forall s \in S: \quad H_{t+1}(s) \leftarrow |E(s; V_{t+1})|. \tag{4}$$

[2] See Appendix A for an example that shows PS may not converge to the optimal value function if some states' priority values are initialized to 0.

Note that Andre et al. (1998) also provide a heuristic approach to updating $H(s)$ without explicitly computing Bellman errors in all states. We only consider GenPS that always maintains the exact absolute Bellman errors.[3] As shown by Lemma 1 in the next subsection, this condition can be satisfied quite efficiently without recomputing Bellman errors for all states in every step.

2.2 Do Prioritized-Sweeping Algorithms Converge?

Although prioritized-sweeping algorithms have consistently enjoyed empirical success for improved convergence rate (Moore & Atkeson, 1993; Peng & Williams, 1993; Andre et al., 1998; Wingate & Seppi, 2005), it remains an open question whether their value-function estimates, V_t, will converge to V^* in the limit. Moore and Atkeson (1993) conjectured that such a proof might rely on convergence results in ADP by observing that PS is an ADP algorithm.

It is known that an ADP algorithm can converge to V^* provided that it performs Bellman backups in *every* state *infinitely* often—that is, no state is *starved* of updates, which is obviously not always guaranteed for prioritized-sweeping algorithms. Take GenPS as an example because of its conceptual simplicity. Andre et al. (1998) propose to overestimate priorities as a way to avoid starving states of updates. But, this modification may still starve *other* states, as shown by the example in Appendix B. Similarly, Wingate and Seppi (2005) argue that every state with nonzero Bellman error will eventually be updated in GenPS, but do not justify this key claim. This claim would follow if after every Bellman backup, the "total" Bellman error, evaluated somehow over all states, decreases. Synchronous value iteration, for example, has this property—convergence follows because the maximum Bellman error over all states decreases each iteration (Puterman, 1994). If it were the case, then a state with nonzero Bellman error would eventually get its value updated, and a convergence proof followed immediately from the general convergence theorem for ADP algorithms. Unfortunately, this conjectured property is not true for most commonly used metrics of Bellman errors.

To show that Bellman backups may not result in steady progress in overall Bellman error, let us take a closer look at how Bellman errors change when a Bellman backup is performed in state s_t at step t. The following lemma will also be useful in our convergence proof later. For convenience, define $E^a(s;V)$ for all $a \in A$ as follows:

$$E^a(s;V) = R(s,a) + \gamma \sum_{s' \in S} T(s'|s,a)V(s') - V(s).$$

It follows from the definition that

$$E(s;V) = \max_{a \in A} E^a(s;V). \tag{5}$$

[3]See Appendix B for an example that shows the heuristic GenPS algorithm may not converge to the optimal value function.

Lemma 1. *Using the notation in Algorithm 1, we have for all $t = 1, 2, \ldots$ that:*

$$E(s; V_{t+1}) = \begin{cases} \max_{a \in A} \left[E^a(s; V_t) + \gamma T(s_t|s,a) E(s_t; V_t) \right], & \text{if } s \neq s_t \\ \max_{a \in A} \left[E^a(s; V_t) + \gamma T(s_t|s_t,a) E(s_t; V_t) \right] - E(s_t; V_t), & \text{if } s = s_t \end{cases}. \quad (6)$$

Proof. We consider two cases separately. For $s \neq s_t$ and any $a \in A$,

$$\begin{aligned} E^a(s; V_{t+1}) &= R(s,a) + \gamma \sum_{s' \in S} T(s'|s,a) V_{t+1}(s') - V_{t+1}(s) \\ &= R(s,a) + \gamma \sum_{s' \in S} T(s'|s,a) V_t(s') + \gamma T(s_t|s,a) E(s_t; V_t) - V_t(s) \\ &= E^a(s; V_t) + \gamma T(s_t|s,a) E(s_t; V_t), \end{aligned}$$

where the first equality is due to the definition of E^a, the second equality follows from the fact that $V_{t+1}(s')$ differs from $V_t(s')$ only when $s' = s_t$ and the difference is exactly $E(s_t; V_t)$, and the last equality is again due to the definition of E^a. Using Eqn. 5, we have proved the first part of Eqn. 6. For $s = s_t$ and any $a \in A$, similar reasoning follows:

$$\begin{aligned} E^a(s_t; V_{t+1}) &= R(s_t,a) + \gamma \sum_{s' \in S} T(s'|s_t,a) V_{t+1}(s') - V_{t+1}(s_t) \\ &= R(s_t,a) + \gamma \sum_{s' \in S} T(s'|s_t,a) V_t(s') + \gamma T(s_t|s_t,a) E(s_t; V_t) - V_t(s_t) - E(s_t; V_t) \\ &= E^a(s_t; V_t) + \gamma T(s_t|s_t,a) E(s_t; V_t) - E(s_t; V_t). \end{aligned}$$

Using Eqn. 5 again, we have proved the second part of Eqn. 6. □

In the special case of uncontrolled MDPs where $A = \{a_1\}$, Eqn. 6 can be simplified to

$$E(s; V_{t+1}) = \begin{cases} E(s; V_t) + \gamma T(s_t|s,a_1) E(s_t; V_t), & \text{if } s \neq s_t \\ \gamma T(s_t|s_t,a_1) E(s_t; V_t), & \text{if } s = s_t \end{cases}.$$

Therefore, the absolute Bellman error in state s_t is guaranteed to decrease, but the same may not be true for other states. The following example that both of the following can happen after a Bellman backup in uncontrolled MDPs:

- The ℓ_∞-norm Bellman error, defined by $\max_{s \in S} |E(s; V_t)|$, may increase.
- The ℓ_1-norm Bellman error, defined by $\sum_{s \in S} |E(s; V_t)|$, may increase.

Example 1. *Figure 1 gives a deterministic, uncontrolled Markov decision process, which has $n+1$ state: $S = \{s^0, s^1, \cdots, s^n\}$. Assume the present value function, V_t, is zero for all states. All states then have*

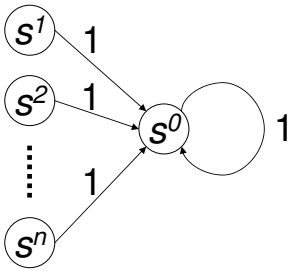

Figure 1: A Deterministic, Uncontrolled Markov Decision Process. The transitions are deterministic and are indicated by arrows that are labeled by immediate rewards.

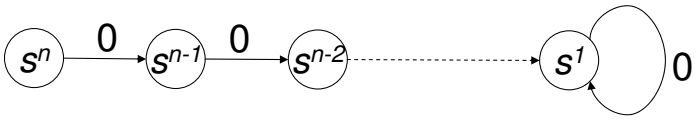

Figure 2: A Deterministic, Uncontrolled Markov Decision Process. The transitions are deterministic and are indicated by arrows that are labeled by immediate rewards.

the same Bellman error, 1*, and thus the same priority value. Let a tie be broken so that* s^0 *is chosen for a Bellman backup.*[4] *After the backup, the value function becomes,*

$$V_{t+1}(s) = \begin{cases} 0, & \text{if } s \neq s^0 \\ 1, & \text{if } s = s^0 \end{cases},$$

and the Bellman errors are

$$E(s; V_{t+1}) = \begin{cases} 1+\gamma, & \text{if } s \neq s^0 \\ \gamma, & \text{if } s = s^0 \end{cases}.$$

Therefore, the ℓ_∞*-norm Bellman error is increased from* 1 *to* $1+\gamma$*, and the* ℓ_1*-norm Bellman error is increased from* $n+1$ *to* $n+(n+1)\gamma$*, since* γ *is often close to* 1*.*

Furthermore, we could verify that the value-function estimation error, defined by $V(s) - V^*(s)$, may increase in *any* norm, including the ℓ_∞-norm or the ℓ_1-norm, as is shown in the next example.

Example 2. *Figure 2 depicts a deterministic, n-state uncontrolled MDP in which every reward is* 0*. The true optimal value function is* $V^*(s) = 0$ *for all* $s \in S$*. Suppose the present value function is*

$$V_t(s) = \begin{cases} 0, & \text{if } s = s^n \\ i, & \text{if } s = s^i \end{cases}.$$

[4] Or we could increase the immediate reward of s^0 by a tiny amount so that $H_t(s^0)$ is strictly larger than the other states.

Since the absolute Bellman error in state s^1 is largest, GenPS picks it for a Bellman backup, yielding the following value function:

$$V_{t+1}(s) = \begin{cases} \gamma(n-1), & \text{if } s = s^n \\ i, & \text{if } s = s^i \end{cases}.$$

Therefore, with the value-function error unchanged in all other states, the error in state s^n is increased from 0 to $\gamma(n-1)$.

The previous two examples illustrate that a Bellman backup in a single state may not guarantee the value function estimate is closer to the true value function, either in the sense of smaller Bellman error or smaller value-function error. In fact, we show in Appendix A that PS may not converge to the optimal value function when it initializes $H(s)$ inappropriately. Despite these difficulties, we are able to provide a convergence result in the next section that applies to a large class of prioritized-sweeping algorithms, including PS (under an easy-to-satisfy assumption) and GenPS.

3 Convergence of Prioritized-Sweeping Algorithms

We start with a general convergence result for prioritized-sweeping algorithms, and then apply it to GenPS and PS.

3.1 A General Convergence Lemma

The following key lemma gives a set of sufficient conditions for the convergence of Algorithm 1.

Lemma 2. *Let $F \subseteq S$ be the set of states s that are chosen for Bellman backups infinitely often during the whole run of Algorithm 1; so, $S \setminus F$ consists of states that become starved of backups. Let τ be the* last *step in which some state in $S \setminus F$ is chosen for update. Clearly, $F \neq \emptyset$ by the Pigeonhole Principle. Then the Bellman errors of infinitely updated states are driven to 0 in the limit; formally,*

$$\lim_{t\to\infty} \max_{s\in F} |E(s; V_t)| = 0. \tag{7}$$

Furthermore, the value function V_t converges to V^* if the following conditions hold:

1. *The priority values converge to 0 in the limit:*

$$\lim_{t\to\infty} H_t(s_t) = 0.$$

2. *There exists a constant $C > 0$ such that: $H_t(s) \geq C \cdot |E(s; V_t)|$ for all states $s \notin F$ and $t > \tau$;*

Some intuitions are helpful before presenting the formal proof. The first condition requires that, in the limit, all priority values must approach 0 so that no state with a nonzero, constant priority value will be starved of updates. The second condition requires that, in the limit, the priority value for starving

states *must not be too small* compared to their absolute Bellman errors. Therefore, these two conditions together guarantee that any starving state must have a zero Bellman error, and thus do not need Bellman backups at all.

Proof. Given any MDP $M = \langle S, A, T, R, \gamma \rangle$, we run Algorithm 1 forever using an arbitrary bounded initial value function V_1. Now, we will construct an induced MDP $M' = \langle S, A, T', R', \gamma \rangle$ with the same states, actions, and discount factor. For states $s \in F$, the reward function and transition probabilities are the same as M; for states $s \notin F$,

$$\begin{aligned} \forall a \in A: \quad & R'(s,a) = V_{\tau+1}(s) \cdot (1-\gamma) \\ \forall a \in A: \quad & T'(s'|s,a) = \mathbb{I}(s' = s), \end{aligned}$$

where $\mathbb{I}$ is the set-indicator function. In other words, we turn states $s \notin F$ to absorbing states with self loops, and the new reward is constructed so that $V_{\tau+1}(s)$ is exactly the optimal value of s in M'.

After step τ, Algorithm 1 running on M can be viewed as Algorithm 1 running on M' since states $s \notin F$ will not get their values updated and already have the optimal values in M' anyway. By construction of M', the value function over these states converges to the optimal value function of M' (Bertsekas & Tsitsiklis, 1989), since all states $s \in F$ are updated infinitely often. Hence, the Bellman error evaluated using the model of M' converges to 0 for all states $s \in S$.

Finally, we consider the Bellman error evaluated using the model of M. For states $s \in F$, the Bellman error in s converges to 0, as M and M' have the same dynamics for these states. Thus, we have proved the first part of the lemma:

$$\lim_{t \to \infty} \max_{s \in F} |E(s; V_t)| = 0.$$

We now consider Bellman errors of states outside of F. For any $t > \tau$, we have

$$\max_{s \notin F} |E(s; V_t)| \;\leq\; \max_{s \notin F} \frac{H_t(s)}{C} \;\leq\; \frac{H_t(s_t)}{C},$$

where the first inequality follows from Condition 2, and the second from the operations of Algorithm 1. But Condition 1 requires $H_t(s_t)$ converge to 0, implying that,

$$\lim_{t \to \infty} \left\{ \max_{s \notin F} |E(s; V_t)| \right\} = 0.$$

In other words, the Bellman errors of starving states also converge to 0. Therefore, we have shown that $\lim_{t \to \infty} |E(s; V_t)| = 0$ for *all* states $s \in S$, and thus completed the asymptotic convergence proof since zero Bellman error implies an exact optimal value function. □

3.2 Convergence of GenPS and PS

Lemma 2 is general enough to be applied to a wide class of prioritized-sweeping algorithms. For instance, its two conditions are satisfied by asynchronous value iteration with the priority function defined in Eqn. 1

and Eqn. 2, thus providing yet another convergence proof for this algorithm. As less trivial examples, we prove convergence of GenPS and PS. We start with the simpler one for GenPS.

Theorem 1. *GenPS of Andre et al. (1998) converges to the optimal value function in the limit.*

Proof. Since $H_t(s) = |E(s; V_t)|$ for all $s \in S$ and all $t \geq 1$ in GenPS, Condition 1 of Lemma 2 follows immediately from Eqn. 7, and Condition 2 is satisfied with constant $C = 1$. Therefore, GenPS converges to the optimal value function, by Lemma 2. □

Theorem 2. *PS of Moore and Atkeson (1993) converges to the optimal value function in the limit, if the initial priority values are non-zero, namely, $H_1(s) > 0$ for all $s \in S$.*

Proof. We first verify that Condition 1 of Lemma 2 holds in PS. For $t > \tau$, let t' be the last time s_t is chosen for a Bellman backup.[5] It follows from Eqn. 3 that state s_t's priority value at time t, $H_t(s_t)$, can be written equivalently as:

$$H_t(s_t) = \max_{t' \leq t_0 < t} \left[|E(s_{t_0}; V_{t_0})| \max_a [T(s_{t_0}|s_t, a)] \right].$$

Since the right-hand side above converges to 0, due to Eqn. 7, $H_t(s_t)$ also converges to 0. Thus, Condition 1 of Lemma 2 is satisfied.

Verifying Condition 2 is more involved. Let s be any state outside of F. We first consider the case of $s = s_t$ for some $t \leq \tau$. On one hand,

$$\begin{aligned}
E(s; V_{t+1}) &= \max_{a \in A} [E^a(s; V_t) + \gamma T(s_t|s_t, a) E(s_t; V_t)] - E(s_t; V_t) \\
&\leq \max_{a \in A} [E^a(s; V_t)] + \gamma \max_{a \in A} [T(s_t|s_t, a) E(s_t; V_t)] - E(s_t; V_t) \\
&= \gamma \max_{a \in A} [T(s_t|s_t, a) E(s_t; V_t)] \\
&\leq \gamma \max_{a \in A} [T(s_t|s_t, a)] \, |E(s_t; V_t)| \\
&= \gamma H_{t+1}(s_t),
\end{aligned}$$

where the first equality is justified by Lemma 1, the first inequality is due to the fact that $\max_x [f(x) + g(x)] \leq \max_x f(x) + \max_x g(x)$, the second equality is due to Eqn. 5 and $s = s_t$, and the last equality is due to Eqn. 3. On the other hand, let $a_t = \arg\max_{a \in A} E^a(s_t; V_t)$, then we have

$$\begin{aligned}
E(s; V_{t+1}) &= \max_{a \in A} [E^a(s; V_t) + \gamma T(s_t|s_t, a) E(s_t; V_t)] - E(s_t; V_t) \\
&\geq E^{a_t}(s; V_t) + \gamma T(s_t|s_t, a_t) E(s_t; V_t) - E(s_t; V_t) \\
&= \gamma T(s_t|s_t, a_t) E(s_t; V_t) \\
&\geq -\gamma T(s_t|s_t, a_t) \, |E(s_t; V_t)| \\
&\geq -\gamma \max_{a \in A} [T(s_t|s_t, a)] \, |E(s_t; V_t)| \\
&= -\gamma H_{t+1}(s_t),
\end{aligned}$$

[5] Since s_t is chosen for backups infinitely often if $t > \tau$, we can always find a t large enough so that t' exists.

where the first equality is again due to Lemma 1, the second equality is by definition of a_t and $s = s_t$, and the last equality is due to Eqn. 3. Therefore, we have proved for all t that $|E(s_t; V_{t+1})| \leq \gamma H_{t+1}(s_t)$, or equivalently,

$$H_{t+1}(s_t) \geq \frac{1}{\gamma} |E(s_t; V_{t+1})| . \tag{8}$$

In general, a similar analysis for $s \neq s_t$ seems impossible. So, we use another trick. For any state $s \notin F$, let $\tau_s \leq \tau$ be the *last* time that it is chosen for update by PS.[6] Due to Eqn. 8, we have

$$H_{\tau_s+1}(s) \geq \frac{1}{\gamma} |E(s; V_{\tau_s+1})| . \tag{9}$$

As verified above, $H_t(s)$ converges to 0 in the limit, and it is non-decreasing for $t > \tau_s$, due to Eqn. 3. Therefore, $H_{\tau_s+1}(s) = 0$, implying $E(s; V_{\tau_s+1}) = 0$. Furthermore, we can show that $E(s; V_t) = 0$ for all $t > \tau_s$. Otherwise, let t be the first time after τ_s in which $E(s; V_t) \neq 0$. This can happen only when the Bellman backup on s_t at step t makes the Bellman error of s be nonzero; namely, $|E(s_t; V_t)| \max_{a \in A}[T(s|s_t, a)] \neq 0$. But this also causes $H_{t+1}(s) \neq 0$, contracting the fact that $H_t(s) = 0$ for *all* $t > \tau_s$. Since both $H_t(s)$ and $E_t(s; V_t)$ are zero, Condition 2 is satisfied trivially with constant $C = 1$.

Thus, PS converges to the optimal value function, due to Lemma 2. □

The above theorem requires $H_1(s) > 0$ for all states s, which guarantees that every state is chosen for Bellman backup for at least once in the proof and so τ in Eqn. 9 is well-defined. Appendix A gives an example showing that this condition is necessary: if a state starts with a zero priority value, its state value may never be updated, and thus does not converge to the optimal value.

4 Applications to Model-based Reinforcement Learning

We briefly motivate the use of our convergence results to obtain PAC-MDP model-based reinforcement-learning algorithms (Strehl, Li, & Littman, 2006) that uses prioritized-sweeping algorithms as internal planners.

- An obvious use of prioritized-sweeping algorithms is to solve for (near-)optimal value functions in model-based algorithms such as Rmax (Brafman & Tennenholtz, 2002). The convergence (to the optimal value function) result we established implies the same theoretical performance guarantees (compared to using plain value iteration for planning) while having the potential computational benefits in practice.

[6] We have verified Condition 1 of Lemma 2, which requires that the priority values of non-starving states converge to 0 in the limit. Therefore, any state s will be chosen for Bellman backup at least once if $H_1(s) > 0$, since $H_t(s)$ is non-decreasing w.r.t. t if s is *never* chosen for Bellman backup. This justifies the well-define-ness of τ_s.

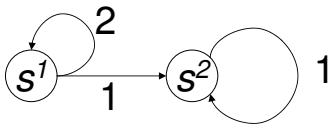

Figure 3: A Two-State Uncontrolled Markov Decision Process. The transitions are indicated by arrows that are labeled by immediate rewards. A transition from s^1 has equal probabilities of reaching s^1 and s^2, while a transition from s^2 always comes back to the same state.

- A more practical use of prioritized-sweeping algorithms is probably in incremental model-based algorithms such as RTDP-Rmax (Strehl et al., 2006). Here, instead of running the planning algorithm until convergence, which is a very expensive operation, we may do "light-weighted" planning by performing only a limited number of Bellman backups in every online timestep. By a similar analysis, we can obtain essentially the same theoretical performance guarantees while expecting these algorithms to learn faster in practice than RTDP-Rmax. This idea is similar to Dyna (Sutton, 1990) where *relaxation planning* is used to accelerate convergence.
- It would be interesting to combine prioritized-sweeping algorithms with function approximation. A safe way, for example, is to combine it with fitted value iteration that is guaranteed to convergence.

A A Counterexample for PS with Zero Initial Priority Values

This section gives a two-state uncontrolled MDP (Figure 3) in which the PS algorithm using zero initial priority values may not converge to the optimal value function. The MDP's state space is $S = \{s^1, s^2\}$, and the optimal value function is

$$V^*(s^1) = \frac{3 - 2\gamma}{(1-\gamma)(2-\gamma)}, \quad V^*(s^2) = \frac{1}{1-\gamma}.$$

Assume the state values are initialized to zero, and the priority values are initialized so that $H_1(s^1) > 0$ and $H_1(s^2) = 0$. We claim that setting $H_1(s^2) = 0$ prevents PS from converging to the optimal value function. In fact, s^1 will be chosen by PS for the first Bellman backup since $H_1(s^1) > H_1(s^2)$, yielding the following value function

$$V_2(s^1) = 1, \quad V_2(s^2) = 0,$$

and priority values

$$H_2(s^1) = 0.5, \quad H_2(s^2) = 0.$$

Then, s^1 will be chosen for the second Bellman backup since $H_2(s^1) > H_2(s^2)$. It can be verified that $H_t(s^1) > 0$ and $H_t(s^2) = 0$ for all $t \geq 1$, and thus $V_t(s^2)$ is updated. Finally, PS converges to the following solution:

$$\hat{V}(s^1) = \frac{2}{2-\gamma}, \quad \hat{V}(s^2) = 0,$$

which does not equal V^* in either state.

B A Counterexample for Heuristic GenPS

This section uses the same uncontrolled MDP in Figure 3 to show that the heuristic version of GenPS, given by Andre et al. (1998), may not converge to the optimal value function. Instead of using Eqn. 4, this algorithm updates the priority value as follows:

$$\forall s \in S: \quad H_{t+1}(s) \leftarrow H_t(s) + \Delta_t \cdot \max_{a \in A} T(s_t|s,a)$$

where $\Delta_t = |V_{t+1}(s_t) - V_t(s_t)| = |E(s_t; V_t)|$ is the change of s_t's value after the most recent Bellman backup. It can be shown that $H_t(s)$ always overestimates $|E_t(s)|$. But this property does not guarantee convergence to the optimal value function, as we will show.

Again, assume the state values are initialized to zero, and the priority values are initialized to the exact absolute Bellman error: $H_1(s^1) = 1.5$ and $H_1(s^2) = 1$. Since $H_1(s^1) > H_1(s^2)$, s^1 will be chosen for the first Bellman backup, yielding the following value function

$$V_2(s^1) = 1, \quad V_2(s^2) = 0,$$

and priority values

$$H_2(s^1) = 2, \quad H_2(s^2) = 1.$$

Then, s^1 will be chosen for the second Bellman backup since $H_2(s^1) > H_2(s^2)$. It can be verified that $H_t(s^1)$ keeps increasing, while $H_t(s^2)$ remains to be 1 and thus $V_t(s^2)$ is never updated. Finally, the algorithm converges to the following solution:

$$\hat{V}(s^1) = \frac{2}{2-\gamma}, \quad \hat{V}(s^2) = 0,$$

which does not equal V^* in either state.

References

Andre, D., Friedman, N., & Parr, R. (1998). Generalized prioritized sweeping. In *Advances in Neural Information Processing Systems 10*, pp. 1001–1007.

Bellman, R. (1957). *Dynamic Programming*. Princeton University Press.

Bertsekas, D. P., & Tsitsiklis, J. N. (1989). *Parallel and Distributed Computation: Numerical Methods*. Prentice Hall. Rrepublished by Athena Scientific in 1997.

Brafman, R. I., & Tennenholtz, M. (2002). R-max—a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, *3*, 213–231.

Moore, A. W., & Atkeson, C. G. (1993). Prioritized sweeping: Reinforcement learning with less data and less time. *Machine Learning*, *13*(1), 103–130.

Peng, J., & Williams, R. J. (1993). Efficient learning and planning within the Dyna framework. In *Proceedings of the Second International Conference on Simulation of Adaptive Behavior*, pp. 281–290.

Puterman, M. L. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley-Interscience, New York.

Strehl, A. L., Li, L., & Littman, M. L. (2006). Incremental model-based learners with formal learning-time guarantees. In *Proceedings of the Twenty-Second Conference on Uncertainty in Artificial Intelligence (UAI-06)*.

Sutton, R. S. (1990). Integrated architectures for learning, planning and reacting based on approximating dynamic programming. In *Proceedings of the Seventh International Conference on Machine Learning (ICML-90)*, pp. 216–224.

Wingate, D., & Seppi, K. D. (2005). Prioritization methods for accelerating MDP solvers. *Journal of Machine Learning Research*, *6*, 851–881.