A New Optimal Stepsize for Approximate Dynamic Programming

Ilya O. Ryzhov, *Member, IEEE*, Peter I. Frazier, and Warren B. Powell, *Member, IEEE*

***Abstract*—Approximate dynamic programming (ADP) has proven itself in a wide range of applications spanning large-scale transportation problems, health care, revenue management, and energy systems. The design of effective ADP algorithms has many dimensions, but one crucial factor is the stepsize rule used to update a value function approximation. Many operations research applications are computationally intensive, and it is important to obtain good results quickly. Furthermore, the most popular stepsize formulas use tunable parameters and can produce very poor results if tuned improperly. We derive a new stepsize rule that optimizes the prediction error in order to improve the short-term performance of an ADP algorithm. With only one, relatively insensitive tunable parameter, the new rule adapts to the level of noise in the problem and produces faster convergence in numerical experiments.**

***Index Terms*—Approximate dynamic programming (ADP), Kalman filter, simulation-based optimization, stochastic approximation.**

I. Introduction

APPROXIMATE dynamic programming (ADP) has emerged as a powerful tool for solving stochastic optimization problems in inventory control [1], emergency response [2], health care [3], energy storage [4]–[6], revenue management [7], and sensor management [8]. In recent research, ADP has been used to solve a large-scale fleet management problem with 50,000 variables per time period and millions of dimensions in the state variable [9], and an energy resource planning problem with 175,000 time periods [10]. Applications in operations research are especially demanding, often requiring the sequential solution of linear, nonlinear or integer programming problems. When an ADP algorithm is limited to a few hundred iterations, it is important to find a good solution as quickly as possible, a process that hinges on a stepsize (or learning rate) which controls how new information is merged with existing estimates.

Manuscript received March 15, 2014; revised July 15, 2014 and July 16, 2014; accepted August 30, 2014. Date of publication September 12, 2014; date of current version February 19, 2015. This work was supported in part by AFOSR under contracts FA9550-08-1-0195, FA9550-11-1-0083, and FA9550-12-1-0200, by the National Science Foundation (NSF) under contracts CMMI-1254298, IIS-142251, and IIS-1247696, and by ONR under contract N00014-07-1-0150 through the Center for Dynamic Data Analysis. Recommended by Associate Editor C. Szepesvári.

I. O. Ryzhov is with the Robert H. Smith School of Business, University of Maryland, College Park, MD 20742 USA (e-mail: iryzhov@rhsmith.umd.edu).

P. I. Frazier is with the Department of Operations Research and Information Engineering, Cornell University, Ithaca, NY 14853 USA.

W. B. Powell is with the Department of Operations Research and Financial Engineering, Princeton University, Princeton, NJ 08544 USA.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TAC.2014.2357134

We illustrate the learning process using the language of classical Markov decision processes. Consider an infinite-horizon dynamic program where $V(S)$ is the value of being in state $S \in \mathcal{S}$ and $C(S,x)$ is a reward earned from being in state S and taking action $x \in \mathcal{X}$. It is well-known [11], [12] that we can find the optimal, infinite-horizon value of each state using value iteration, which requires iteratively computing, for each state $S \in \mathcal{S}$

$$V^n(S) = \max_{x \in \mathcal{X}} C(S,x) + \gamma \mathbb{E}\left[V^{n-1}\left(S'(S,x,W)\right)|S,x\right] \quad (1)$$

where $\gamma < 1$ is a discount factor and S' is a random variable describing the next state given that we were in state S, took action x, and observed random information $W \in \mathcal{W}$.

There are many problems where (1) is difficult to solve due to the curse of dimensionality. For this reason, there is a tradition, dating back to Bellman's earliest work [13], of solving this equation approximately. This field has evolved under a variety of names including approximate dynamic programming (ADP), neuro-dynamic programming, and reinforcement learning [see [14]–[17]]. In one major class of ADP methods known as *approximate value iteration*, an observation of the value $V(S)$ is bootstrapped from an approximation of the downstream value of S', and then used to update that approximation. A generic procedure for computing the observation is given by

$$\hat{v}^n = \max_{x^n} \sum_{w \in \mathcal{W}} P(W^{n+1} = w | S^n, x^n) \left[C(S^n, x^n, w) + \gamma \bar{V}^{n-1}\left(S^{n+1}(S^n, x^n, w)\right)\right] \quad (2)$$

where $\bar{V}^{n-1}$ is the value function approximation, and $S^n \in \mathcal{S}$ is our state during the nth iteration of the ADP algorithm. We intend (2) only to illustrate the concept of constructing $\hat{v}^n$ from $\bar{V}^{n-1}$; in practice, the summation in (2) is also approximated. The expectation within the max operator can be avoided using the concept of the post-decision state [17], [18], which enables us to compute a modified version of (2) exceptionally quickly. This makes approximate value iteration particularly useful for online applications (that is, those run in the field), since it is very easy to implement.

Regardless of the particular technique used, we update $\bar{V}^{n-1}$ by smoothing it with the new observation $\hat{v}^n$, obtaining

$$\bar{V}^n(S^n) = (1 - \alpha_{n-1})\bar{V}^{n-1}(S^n) + \alpha_{n-1}\hat{v}^n \quad (3)$$

where $0 < \alpha_{n-1} \leq 1$ is a stepsize (or learning rate). Note again that (3) uses statistical bootstrapping, where the estimate of the value $\hat{v}^n$ depends on a statistical approximation $\bar{V}^{n-1}(S^n)$.

This is the defining characteristic of approximate value iteration, which has proven to be very successful in broad classes of operations research applications. The reinforcement learning community uses a closely related algorithm known as Q-learning [19], which uses a similar bootstrapping scheme to learn the value of a state-action pair. In both approximate value iteration and Q-learning, the stepsize plays two roles. First, it smooths out the effects of noise in our observations (the lower the stepsize, the smoother the approximation). Second, it determines how much weight is placed on new rewards (the higher the stepsize, the more a new reward is worth). This dual role of the stepsize is specific to bootstrapping-based methods, whose ease of use makes them a natural approach for large-scale applications in operations research where rate of convergence is crucial. See e.g., [20] or ch. 14 of [17] for more examples of such applications. The method of using a stepsize to update the value of a state-action pair is based on the field of stochastic approximation; see [21] and [22] for thorough treatments of this field. [23] and [24] were the first to apply this theory to show the convergence of an ADP algorithm when the stepsize rule satisfies certain conditions.

In general, ADP practice shows a strong bias toward simple rules that are easy to code. One example of such a stepsize rule is $\alpha_{n-1} = 1/n$, which has the effect of averaging over the observations $\hat{v}^n$. In fact, this rule satisfies the necessary theoretical conditions for convergence, which has made it into a kind of default rule (see, e.g., [25] for a recent example). However, the literature has acknowledged [26]–[28] that the $1/n$ rule can produce very slow convergence; one of our contributions in this paper is to derive new theoretical bounds providing insights into the weakness of this rule. For this reason, many practitioners use a simple constant stepsize such as $\alpha_{n-1} \equiv 0.1$. Nonetheless, the constant stepsize can produce slow initial convergence for some problems and volatile, non-convergent estimates in the limit. It is also easy to construct problems where any single constant will work poorly. Reference [29] solves an inventory problem for spare parts, where a high-volume spare part may remain in inventory for just a few days, while a low-volume part may remain in inventory for hundreds of days. A small stepsize will work very poorly with a low-volume part, while large stepsizes fail to dampen the noise, and are not appropriate for high-volume parts.

Such applications require the use of stochastic stepsize rules, where α_{n-1} is computed adaptively from the error in the previous prediction or estimate. These methods include the stochastic gradient rule of [30] (and other stochastic gradient algorithms, e.g., by [31] and [32]), the Delta-Bar-Delta rule of [33] and its variants [34], and the Kalman filter [35], [36]. A detailed survey of both deterministic and stochastic stepsizes is given in [37], with additional references in [34]. The main challenge faced by these methods is that the prediction error is difficult to estimate in a general MDP, often resulting in highly volatile stepsizes with large numbers of tunable parameters. A recent work by [38] adopts a different approach based on the relative frequency of visits to different states, but is heavily tied to on-policy learning, whereas practical implementations often use off-policy learning to promote exploration [39]. In all of these cases, the literature largely ignores the dependence of the observation $\hat{v}^n$ on the previous value function approximation $\bar{V}^{n-1}$, arguably the defining feature of approximate value iteration. For instance, the OSA algorithm of [37], which can be viewed as a bias-adjusted Kalman filter (or BAKF, the name used in [17]), assumes independent observations.

We approach the problem of stepsize selection by studying an MDP with a single state and action. This model radically streamlines the behavior of a general DP, but retains key features of DP problems that are crucial to stepsize performance, namely the bias-variance tradeoff and the dependence of observations. We use this model to make the following contributions: 1) We derive easily computable, convergent upper and lower bounds on the time required for convergence under $1/n$, demonstrating that the rate of convergence of $1/n$ can be so slow that this rule should almost never be used for ADP. 2) We derive a closed-form, easily computable stepsize rule that is optimal for the single-state, single-action problem. This is the first stepsize rule to account for the dependence of observations in ADP. The formula requires no tuning, and is easy to apply to a general multi-state problem. 3) We analyze the convergence properties of our stepsize rule. We show that it does not stall, and declines to zero in the limit. This is the first optimal stepsize for ADP that provably has these properties. 4) We present numerical comparisons to other stepsizes in a general ADP setting and demonstrate that, while popular competing strategies are sensitive to tunable parameters, our new rule is robust and fairly insensitive to its single parameter. This last property is of vital practical importance, allowing developers to focus on approximation strategies without the concern that poor performance may be due to a poorly tuned stepsize formula.

Section II defines the optimality of a stepsize, and illustrates the need for an optimal stepsize rule by theoretically demonstrating the poor performance of $\alpha_{n-1} = 1/n$ on our single-state, single-action problem. Section III derives the optimal stepsize rule for the approximate value iteration problem, and shows how it can be used in a more general ADP setting. Section IV presents a numerical sensitivity analysis of the new rule in the single-state problem. Finally, Sections V–VI present numerical results for more general ADP examples.

II. Setup and Motivation

Section II-A lays out the stylized ADP model used for our analysis, and defines the optimality of a stepsize in this setting. Section II-B motivates the need for an optimal stepsize by showing that the commonly used stepsize $\alpha_{n-1} = 1/n$ produces unusably slow convergence in our model.

A. Mathematical Model

In the dynamic programming literature, the notion of an "optimal" stepsize most commonly refers to the solution to the optimization problem

$$\min_{\alpha_{n-1}\in[0,1]} \mathbb{E}\left[\left(\mathbb{E}\hat{v}^n - \bar{V}^n(S^n)\right)^2\right]. \tag{4}$$

We refer to the quantity inside the expectation as the *prediction error*. Recall from (2) that $\hat{v}^n$ serves as an observation (albeit an approximate one) of the value of being in state S^n. The

prediction error is the squared difference between this observation and the current estimate $\bar{V}^{n-1}(S^n)$ of the value.

The prediction error is a standard objective for an optimal stepsize rule, and is used in reinforcement learning (e.g., the IDBD algorithm of [33], used by [40] in RL), stochastic gradient methods [30], Kalman filtering [36], and signal processing [37]. The main challenge faced by researchers is that, for a general dynamic program, (4) cannot be solved in closed form. For this reason, most error-minimizing stepsize algorithms (including very recent work in this area; see [34] for an overview) adopt a gradient descent approach, in which the stepsize is adjusted based on an estimate of the derivative of (4) with respect to α_{n-1}. The resulting stepsize algorithms are no longer optimal, and can exhibit volatile behavior in the early stages. Many of them require extensive tuning.

While we also seek to minimize prediction error, we adopt a different approach. Instead of approximating (4) in the general case, we consider a stylized dynamic program with a single state and a single action, where (4) has a closed-form solution. In this setting, (1) reduces to $v^* = c + \gamma v^*$ and has the solution $v^* = c/1 - \gamma$. The ADP equations (2) and (3) reduce to

$$\hat{v}^n = \hat{c}^n + \gamma \bar{v}^{n-1}, \tag{5}$$

$$\bar{v}^n = (1 - \alpha_{n-1})\bar{v}^{n-1} + \alpha_{n-1}\hat{v}^n \tag{6}$$

where the random variables $\hat{c}^n$, $n = 1, 2, \ldots$ are independent and identically distributed, with $c = \mathbb{E}\hat{c}^n$ and $\sigma^2 = Var(\hat{c}^n)$. The prediction error in this setting reduces to the formulation

$$\min_{\alpha_{n-1} \in [0,1]} \mathbb{E}\left[(\mathbb{E}\hat{v}^n - \bar{v}^n)^2 \right].$$

Although the system described by (5) and (6) is much simpler than a general MDP, it nonetheless retains two key features that are fundamental to all DPs:

1) A tradeoff between bias and variance in the approximation $\bar{v}^n$, governed by the stepsize α_{n-1};
2) Dependence of the bootstrapped observation $\hat{v}^n$ on the approximation $\bar{v}^{n-1}$.

In Section III-D, we consider a finite-horizon extension that captures a third key feature:

3) Time-dependence of the bias-variance tradeoff.

Of course, in a general DP, these issues exhibit much more complex behavior than in the streamlined single-state, single-action model. However, the stylized model is still subject to these issues, and can provide insight into how they can be resolved in the general case. The main advantage offered by this model is that it allows us to address these issues using a closed-form solution for the optimal stepsize, explicitly capturing the relationship between the bias-variance tradeoff and the dependence of the observations. We will then be able to adapt the solution of the single-state problem to general dynamic programs (in Section III-C).

We briefly note that we allow the observation $\hat{c}^n$ in (5) to be random. At a very high level, this allows us to view the single-state, single-action problem as a stand-in for an infinite-horizon MDP in steady state. Recall the well-known property of Markov decision processes that a) the policy produced by the basic value iteration update in (1) converges to an optimal policy and b) the probability that we are in some state s converges to a steady-state distribution (see [12]). As a result, the unconditional expectation of the contribution earned at each iteration approaches a constant that we can denote by c. Again, while the single-state, single-action model cannot capture all of the complexity of a general DP, it allows us to distill a large class of DPs into a simple and elegant archetype capturing key behaviors common to that class.

B. Motivation: Slow Convergence of $\alpha_{n-1} = 1/n$

The research on error-minimizing stepsizes is motivated by the poor practical performance of simple stepsize rules. Among these, the most notable is $\alpha_{n-1} = 1/n$, which produces provably convergent estimates of the value function [23], and thus persists in the literature as a kind of default rule, as evidenced by its recent use in, e.g., [25]. The theoretical worst-case convergence rate of this stepsize is known to be slow [27]. We now derive new bounds that are easier to compute and demonstrate that the $1/n$ rule is unusably slow even for the stylized single-state, single-action model of Section II-A.

Consider the ADP model of (5) and (6). For simplicity, we assume in this discussion that $\hat{c}^n = c$ for all n, that is, all the rewards are deterministic. If an algorithm performs badly in this deterministic case, we generally expect it to perform even worse when $\hat{c}^n$ is allowed to be random, since increasing noise generally slows convergence. We briefly summarize our results and give a numerical illustration; the full technical details can be found in the extended version of this paper [41].

Theorem 1: $\bar{v}^n \geq (c/(1-\gamma)(1-(n+1)^{-(1-\gamma)})$ for $n = 0, 1, 2, \ldots$.

Theorem 2: $\bar{v}^n \leq (c/(1-\gamma)[1-bn^{-(1-\gamma)} - (1-\gamma)/\gamma)(1/n)]$ for all $n = 1, 2, \ldots$ where $b = (\gamma^2 + \gamma - 1)/\gamma$.

In our numerical illustration, we fix c to 1, because it only enters as a multiplicative factor in the bounds and in the true value function as well. Thus γ is our only free parameter. The results are plotted on a log-scale in Fig. 1. As n grows large the upper and lower bounds both approach the limiting value $v^* = 1/(1-\gamma)$. Convergence slows as γ increases.

In Fig. 2, we show the number of iterations before $\bar{v}^n$ reaches 1% of optimal. The lower bound on the value of $\bar{v}^n$ gives an upper bound on the number of iterations needed, and the upper bound on $\bar{v}^n$ gives a lower bound on the iterations needed. For γ near .7, we already require 10,000 iterations, causing difficulty for applications requiring a significant amount of time per iteration. Then, as γ grows larger than .8 we require at least 10^8 iterations, which is impractical for almost any application. As γ grows above .9, the number of iterations needed is at least 10^{19}.

We see that, in this simple problem, approximate value iteration with stepsize $1/n$ converges so slowly as to be impractical for most infinite horizon applications, particularly when the discount factor is close to 1. This behavior is likely to be seen in other more complex infinite horizon problems, and also in undiscounted finite horizon problems. The remainder of this paper studies a new stepsize rule that is optimal for the single-state, single-action MDP used in the above analysis.

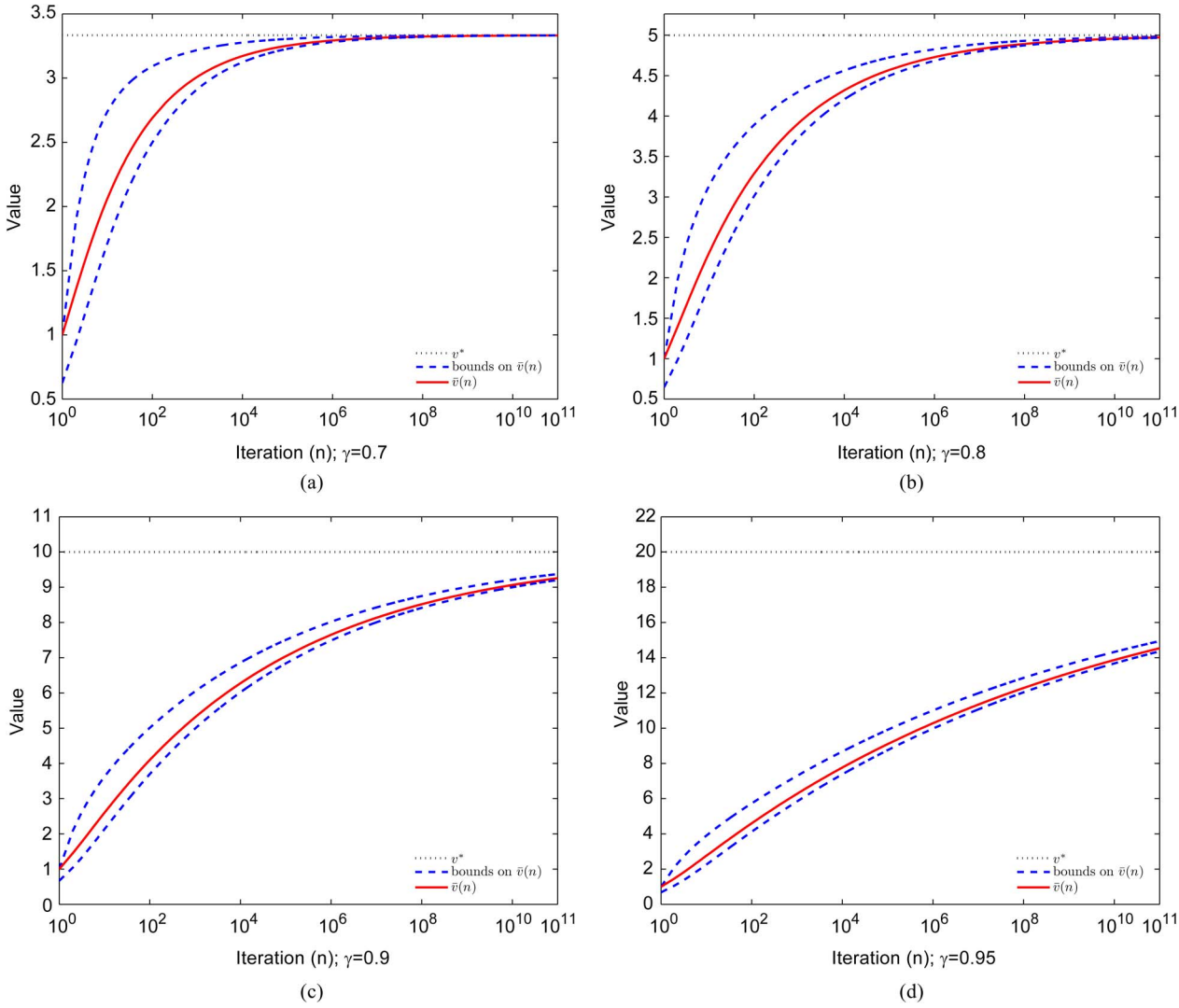


Fig. 1. $\bar{v}(n)$ and its upper and lower bounds for different discount factors; (a) $\gamma = 0.7$; (b) $\gamma = 0.8$; (c) $\gamma = 0.9$; (d) $\gamma = 0.95$.

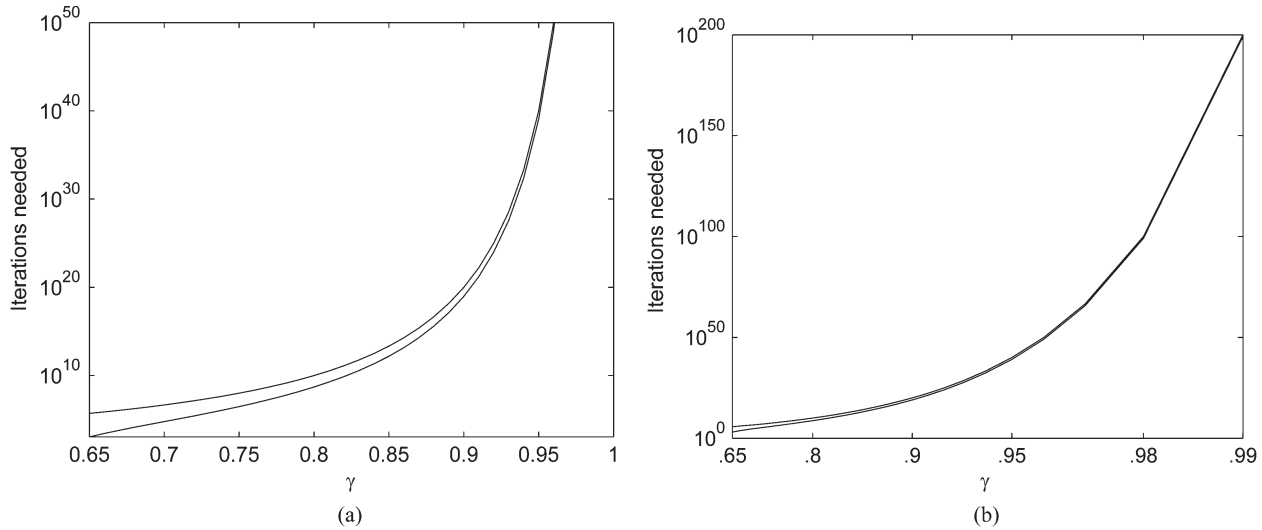


Fig. 2. Upper and lower bounds on number of iterations needed to get within 1% of optimal, plotted for different ranges of γ; (a) $0.65 \leq \gamma \leq 1$; (b) $0.65 \leq \gamma \leq 0.99$.

III. An Optimal Stepsize for Approximate Value Iteration

In Section III-A, we derive a new stepsize rule that is optimal for the approximate value iteration problem given by (5) and (6). We then study its convergence properties in Section III-B. However, while we use the special case in (5)–(6) for theoretical tractability, our ultimate goal is to obtain an algorithm that can be applied in a general dynamic program. This extension is explained in Section III-C, and the general form of our stepsize is given in (20). Finally, Section III-D considers an extension to finite-horizon problems.

A. Derivation

The approximate value iteration problem is given by (5) and (6). As before, let $c = \mathbb{E}\hat{c}^n$ and $\sigma^2 = Var(\hat{c}^n)$. Observe that $\bar{v}^n$ can be written recursively as

$$\begin{aligned}\bar{v}^n &= (1-\alpha_{n-1})\bar{v}^{n-1} + \alpha_{n-1}\hat{c}^n + \alpha_{n-1}\gamma\bar{v}^{n-1} \\ &= \left(1-(1-\gamma)\alpha_{n-1}\right)\bar{v}^{n-1} + \alpha_{n-1}\hat{c}^n. \end{aligned} \tag{7}$$

The particular structure of this problem allows us to derive recursive formulas for the mean and variance of the approximation $\bar{v}^n$. We assume that $\bar{v}^0 = 0$.

Proposition 1: Define

$$\delta^n = \begin{cases} \alpha_0 & n = 1 \\ \alpha_{n-1} + \left(1-(1-\gamma)\alpha_{n-1}\right)\delta^{n-1} & n > 1 \end{cases}$$

and

$$\lambda^n = \begin{cases} \alpha_0^2 & n = 1 \\ \alpha_{n-1}^2 + \left(1-(1-\gamma)\alpha_{n-1}\right)^2\lambda^{n-1} & n > 1. \end{cases}$$

Then, $\mathbb{E}(\bar{v}^n) = \delta^n c$ and $Var(\bar{v}^n) = \lambda^n\sigma^2$.

Proof: Observe that $\mathbb{E}(\bar{v}^1) = \alpha_0 c = \delta^1 c$ and $Var(\bar{v}^1) = \alpha_0^2\sigma^2 = \lambda^1\sigma^2$. Now suppose that $\mathbb{E}(\bar{v}^{n-1}) = \delta^{n-1}c$ and $Var(\bar{v}^{n-1}) = \lambda^{n-1}\sigma^2$. By (7), we have

$$\mathbb{E}(\bar{v}^n) = \left(1-(1-\gamma)\alpha_{n-1}\right)\delta^{n-1}c + \alpha_{n-1}c = \delta^n c.$$

Furthermore, $\bar{v}^{n-1}$ depends only on $\hat{c}^{n'}$ for $n' < n$, therefore $\bar{v}^{n-1}$ and $\hat{c}^n$ are indqependent. Consequently

$$Var(\bar{v}^n) = \left(1-(1-\gamma)\alpha_{n-1}\right)^2\lambda^{n-1}\sigma^2 + \alpha_{n-1}^2\sigma^2 = \lambda^n\sigma^2$$

as required. ■

The next result shows that these quantities are uniformly bounded in n; the proof is given in [41].

Proposition 2: For all n, $\delta^n \le (1/(1-\gamma))$ and $\lambda^n \le (1/\gamma(1-\gamma))$.

We define the optimal stepsize for time n to be the value that achieves

$$\min_{\alpha_{n-1}\in[0,1]} \mathbb{E}\left[\left(\bar{v}^n(\alpha_{n-1}) - \mathbb{E}\hat{v}^n\right)^2\right] \tag{8}$$

which is the minimum squared deviation of the time-n estimate $\bar{v}^n$ from the mean of the time-n observation $\hat{v}^n$. The constraint $\alpha_{n-1} \in [0,1]$ is standard in ADP, but turns out to be redundant here; as we see below (Corollary 4), minimizing the unconstrained objective will produce a solution that always satisfies the constraint.

We can simplify the objective function in (8) in the following manner:

$$\begin{aligned}&\mathbb{E}\left[\left(\bar{v}^n(\alpha_{n-1}) - \mathbb{E}\hat{v}^n\right)^2\right] \\ &\quad= \mathbb{E}\left[\left((1-\alpha_{n-1})\bar{v}^{n-1} + \alpha_{n-1}\hat{v}^n - \mathbb{E}\hat{v}^n\right)^2\right] \\ &\quad= \mathbb{E}\left[\left((1-\alpha_{n-1})(\bar{v}^{n-1} - \mathbb{E}\hat{v}^n) + \alpha_{n-1}(\hat{v}^n - \mathbb{E}\hat{v}^n)\right)^2\right] \\ &\quad= (1-\alpha_{n-1})^2\mathbb{E}\left[\left(\bar{v}^{n-1} - \mathbb{E}\hat{v}^n\right)^2\right] + \alpha_{n-1}^2\mathbb{E}\left[\left(\hat{v}^n - \mathbb{E}\hat{v}^n\right)^2\right] \\ &\qquad+ 2\alpha_{n-1}(1-\alpha_{n-1})\mathbb{E}\left[(\bar{v}^{n-1} - \mathbb{E}\hat{v}^n)(\hat{v}^n - \mathbb{E}\hat{v}^n)\right].\end{aligned}$$

The first equality is obtained using the recursive formula for $\bar{v}^n$ from (7). Observe that

$$\begin{aligned}\mathbb{E}\left[(\bar{v}^{n-1} - \mathbb{E}\hat{v}^n)(\hat{v}^n - \mathbb{E}\hat{v}^n)\right] &= \mathbb{E}(\bar{v}^{n-1}\hat{v}^n) - \mathbb{E}\bar{v}^{n-1}\mathbb{E}\hat{v}^n \\ &= Cov(\bar{v}^{n-1}, \hat{v}^n)\end{aligned}$$

whence we obtain

$$\begin{aligned}&\mathbb{E}\left[\left(\bar{v}^n(\alpha_{n-1}) - \mathbb{E}\hat{v}^n\right)^2\right] \\ &= (1-\alpha_{n-1})^2\mathbb{E}\left[\left(\bar{v}^{n-1} - \mathbb{E}\hat{v}^n\right)^2\right] + \alpha_{n-1}^2\mathbb{E}\left[\left(\hat{v}^n - \mathbb{E}\hat{v}^n\right)^2\right] \\ &\quad+ 2\alpha_{n-1}(1-\alpha_{n-1})Cov(\bar{v}^{n-1}, \hat{v}^n).\end{aligned} \tag{9}$$

The error-minimizing stepsize is unique, due to the convexity of the prediction error; the proof of this property is given in [41].

Proposition 3: The objective function in (8) is convex in α_{n-1}.

Due to Proposition 3, we can solve (8) by setting the derivative of the prediction error equal to zero and solving for α_{n-1}. This yields an equation

$$\begin{aligned}(\alpha_{n-1}-1)\mathbb{E}\left[\left(\bar{v}^{n-1} - \mathbb{E}\hat{v}^n\right)^2\right] + \alpha_{n-1}\mathbb{E}\left[\left(\hat{v}^n - \mathbb{E}\hat{v}^n\right)^2\right] \\ + (1-2\alpha_{n-1})Cov(\bar{v}^{n-1}, \hat{v}^n) = 0\end{aligned}$$

whence we obtain

$$\alpha_{n-1} = \frac{\mathbb{E}\left[\left(\bar{v}^{n-1} - \mathbb{E}\hat{v}^n\right)^2\right] - Cov(\bar{v}^{n-1}, \hat{v}^n)}{\mathbb{E}\left[\left(\bar{v}^{n-1} - \mathbb{E}\hat{v}^n\right)^2\right] + \mathbb{E}\left[\left(\hat{v}^n - \mathbb{E}\hat{v}^n\right)^2\right] - 2Cov(\bar{v}^{n-1}, \hat{v}^n)}. \tag{10}$$

We now present our main result, which gives an explicit formula for (10).

Theorem 3: Assuming that α_0 is given, the optimal time-n stepsize can be computed using the formula

$$\alpha_{n-1} = \frac{(1-\gamma)\lambda^{n-1}\sigma^2 + \left(1-(1-\gamma)\delta^{n-1}\right)^2 c^2}{(1-\gamma)^2\lambda^{n-1}\sigma^2 + \left(1-(1-\gamma)\delta^{n-1}\right)^2 c^2 + \sigma^2} \tag{11}$$

where δ^{n-1} and λ^{n-1} are as in Proposition 1.

Proof: We compute each expectation in (10). First, observe that

$$\mathbb{E}\left[(\hat{v}^n - \mathbb{E}\hat{v}^n)^2\right] = Var(\hat{v}^n) = (1+\gamma^2\lambda^{n-1})\sigma^2$$

using the independence of $\hat{c}^n$ and $\bar{v}^{n-1}$ together with Proposition 1. We now use a bias-variance decomposition (see e.g., [42]) to write

$$\begin{aligned}
&\mathbb{E}\left[\left(\bar{v}^{n-1} - \mathbb{E}\hat{v}^n\right)^2\right] \\
&\quad = \mathbb{E}\left[\left(\bar{v}^{n-1} - \mathbb{E}\bar{v}^{n-1} + \mathbb{E}\bar{v}^{n-1} - \mathbb{E}\hat{v}^n\right)^2\right] \\
&\quad = \mathbb{E}\left[\left(\bar{v}^{n-1} - \mathbb{E}\bar{v}^{n-1}\right)^2\right] + \left(\mathbb{E}\bar{v}^{n-1} - \mathbb{E}\hat{v}^n\right)^2 \\
&\quad = \mathrm{Var}(\bar{v}^{n-1}) + \left(\mathbb{E}\bar{v}^{n-1} - \mathbb{E}\hat{v}^n\right)^2 \qquad (12)
\end{aligned}$$

where the cross term vanishes because the quantity $\mathbb{E}\bar{v}^{n-1} - \mathbb{E}\hat{v}^n$ is deterministic, and thus

$$\begin{aligned}
&\mathbb{E}\left[(\bar{v}^{n-1} - \mathbb{E}\bar{v}^{n-1})(\mathbb{E}\bar{v}^{n-1} - \mathbb{E}\hat{v}^n)\right] \\
&\quad = (\mathbb{E}\bar{v}^{n-1} - \mathbb{E}\hat{v}^n)\mathbb{E}(\bar{v}^{n-1} - \mathbb{E}\bar{v}^{n-1}) \\
&\quad = 0.
\end{aligned}$$

By Proposition 1 $Var(\bar{v}^{n-1}) = \lambda^{n-1}\sigma^2$, and

$$\mathbb{E}\hat{v}^n - \mathbb{E}\bar{v}^{n-1} = c + \gamma\delta^{n-1}c - \delta^{n-1}c = \left(1 - (1-\gamma)\delta^{n-1}\right)c$$

represents the bias of $\bar{v}^{n-1}$ in predicting $\hat{v}^n$. Thus

$$\mathbb{E}\left[\left(\bar{v}^{n-1} - \mathbb{E}\hat{v}^n\right)^2\right] = \lambda^{n-1}\sigma^2 + \left(1 - (1-\gamma)\delta^{n-1}\right)^2 c^2.$$

Finally, we compute

$$\begin{aligned}
&\mathrm{Cov}(\bar{v}^{n-1}, \hat{v}^n) \\
&\quad = \mathbb{E}(\bar{v}^{n-1}\hat{v}^n) - \mathbb{E}\bar{v}^{n-1}\mathbb{E}\hat{v}^n \\
&\quad = \mathbb{E}\left(\bar{v}^{n-1}(\hat{c}^n + \gamma\bar{v}^{n-1})\right) - \mathbb{E}\bar{v}^{n-1}\mathbb{E}(\hat{c}^n + \gamma\bar{v}^{n-1}) \\
&\quad = c\mathbb{E}\bar{v}^{n-1} + \gamma\mathbb{E}\left(\bar{v}^{n-1}\right)^2 - c\mathbb{E}\bar{v}^{n-1} - \gamma\left(\mathbb{E}\bar{v}^{n-1}\right)^2 \\
&\quad = \gamma\mathrm{Var}(\bar{v}^{n-1}) \qquad (13)
\end{aligned}$$

where we use the independence of $\bar{v}^{n-1}$ and $\hat{c}^n$ to obtain the third line. Substituting all of these expressions into (10) completes the proof. ■

Corollary 4: For all n, $\alpha_{n-1} \in [0,1]$.

Proof: The positivity of α_{n-1} is obvious from (11), where both the numerator and denominator are sums of positive terms (it can easily be seen that $\lambda^{n-1} \geq 0$ for all n). To show that $\alpha_{n-1} \leq 1$, first observe that

$$\gamma(1-\gamma)\lambda^{n-1}\sigma^2 \leq \sigma^2$$

by the result of Proposition 2. From this it can easily be shown that

$$(1-\gamma)\lambda^{n-1}\sigma^2 \leq (1-\gamma)^2\lambda^{n-1}\sigma^2 + \sigma^2$$

completing the proof. ■

We see that both the numerator and the denominator of the fraction in (11) include covariance terms. To our knowledge, this is the first stepsize in the literature to explicitly account for the dependence between observations. Furthermore, the formula includes a closed-form expression for the bias $\mathbb{E}\bar{v}^{n-1} - \mathbb{E}\hat{v}^n$, which is balanced against the variance of $\bar{v}^{n-1}$.

We close this section by showing that our formula behaves correctly in special cases. If the rewards we collect are deterministic, then our estimate $\bar{v}^n$ is simply adding up the discounted rewards, and should converge to v^* under the optimal stepsize rule. If the process $\hat{v}^n$ is stationary, i.e., $\gamma = 0$, then $\bar{v}^n$ is simply estimating c, and we should be using the known optimal stepsize rule of $\alpha_{n-1} = 1/n$.

Corollary 5: If the underlying reward process has zero noise, then $\sigma^2 = 0$ and $\alpha_{n-1} = 1$ for all n. It follows that $\bar{v}^n = \hat{v}^n$ for all n:

$$\lim_{n\to\infty} \bar{v}^n = \sum_{i=0}^{\infty} \gamma^i c = \frac{c}{1-\gamma}.$$

Corollary 6: If the problem is stationary, that is, $\gamma = 0$, then the optimal stepsize is given by $\alpha_{n-1} = 1/n$ for all n as long as $\alpha_0 = 1$.

Proof: If $\alpha_1 = 1$ and $\gamma = 0$, then $\hat{v}^n = \hat{c}^n$. It can easily be shown by induction that $\mathbb{E}\bar{v}^n = c$ for all n, which means that $\delta^n = 1$ for all n. Then, (11) reduces to

$$\alpha_{n-1} = \frac{\lambda^{n-1}\sigma^2}{(1+\lambda^{n-1})\sigma^2} = \frac{\lambda^{n-1}}{1+\lambda^{n-1}}.$$

We claim that $\lambda^{n-1} = 1/(n-1)$. It is clearly true that $\lambda^1 = 1$, from which it follows that $\alpha_1 = 1/2$. Now suppose that $\alpha_{n-2} = 1/(n-1)$ and $\lambda^{n-2} = 1/(n-2)$. Then

$$\begin{aligned}
\lambda^{n-1} &= \alpha_{n-2}^2 + (1-\alpha_{n-2})^2\lambda^{n-2} \\
&= \frac{1}{(n-1)^2} + \frac{n-2}{(n-1)^2} \\
&= \frac{1}{n-1}
\end{aligned}$$

and $\alpha_{n-1} = 1/n$, as required. ■

B. Convergence Analysis

It is well-known [11], [12] that, with some regularity assumptions on the underlying stochastic processes, a stochastic approximation algorithm is provably convergent as long as $\alpha_{n-1} \geq 0$ for all n and

$$\sum_{n=1}^{\infty} \alpha_{n-1} = \infty, \qquad \sum_{n=1}^{\infty} \alpha_{n-1}^2 < \infty.$$

We show the first condition by establishing a lower bound on α_{n-1}. The proof is given in [41].

Proposition 4: For all $n \geq 1$, $\alpha_{n-1} \geq (1-\gamma)/n$.

From Proposition 4, it follows that:

$$\sum_{n=1}^{\infty} \alpha_{n-1} \geq (1-\gamma)\sum_{n=1}^{\infty} \frac{1}{n} = \infty$$

satisfying one of the conditions for convergence. The second condition $\sum_{n=1}^{\infty} \alpha_{n-1}^2 < \infty$ can sometimes be relaxed to the

requirement that $\alpha_{n-1} \to 0$. For instance, [22] discusses the sufficiency of this requirement in stochastic approximation problems with bounded observations. See also [43] for recent proofs of convergence with weaker conditions on the stepsizes. We do not show almost sure convergence in this paper, but we do show that $\alpha_{n-1} \to 0$, a condition that is common to the above convergence proofs. While this does not automatically imply a.s. convergence, it does produce convergence in L^2 for the single-state, single-action model.

We begin by showing that the bias term in the stepsize formula converges to zero; the proof is given in [41]. We then prove that $\alpha_{n-1} \to 0$.

Proposition 5: $\lim_{n\to\infty} \delta^n = 1/(1-\gamma)$.

Theorem 7: $\lim_{n\to\infty} \alpha_{n-1} = 0$.

Proof: It is enough to show that $\lambda^n \to 0$ and apply (11) together with Proposition 5. We show that every convergent subsequence of λ^n must converge to zero using a proof by contradiction.

First, suppose that n_k is a subsequence satisfying $\lim_{k\to\infty} \lambda^{n_k} = \ell$. Combining this with Proposition 5, we return to (11) and find

$$\lim_{k\to\infty} \alpha_{n_k} = \frac{(1-\gamma)\ell}{(1-\gamma)^2\ell + 1}.$$

We then return to Proposition 1 and derive

$$\begin{aligned}\lim_{k\to\infty} \lambda^{n_k+1} &= \left[\frac{(1-\gamma)\ell}{(1-\gamma)^2\ell+1}\right]^2 + \left[1 - \frac{(1-\gamma)^2\ell}{(1-\gamma)^2\ell+1}\right]^2 \ell \\ &= \frac{\ell}{(1-\gamma)^2\ell+1}. \end{aligned} \tag{14}$$

It follows from (14) that, if $\ell > 0$, then:

$$\lim_{k\to\infty} \lambda^{n_k+1} < \lim_{k\to\infty} \lambda^{n_k}. \tag{15}$$

By Proposition 2, we know that the sequence $(\lambda^n)_{n=1}^\infty$ is bounded. Therefore, the set of accumulation points for this sequence is closed and bounded. Suppose that

$$\limsup_{n\to\infty} \lambda^n = \lambda^*$$

and that $\lambda^* > 0$. Let n_k be a subsequence with $\lambda^{n_k} \to \lambda^*$. The subsequence $(\lambda^{n_k-1})_{k=1}^\infty$ is bounded, and therefore must contain an additional convergent subsequence, which we denote by m_k. Suppose that $\lim_{k\to\infty} \lambda^{m_k} = \ell$. It must be the case that

$$\lim_{k\to\infty} \lambda^{m_k+1} = \lim_{k\to\infty} \lambda^{n_k} = \lambda^*.$$

This implies that $\ell > 0$, because otherwise (14) would imply that $\lambda^* = 0$. However, it then follows from (15) that $\lambda^* < \ell$. This is impossible, because we took λ^* to be the largest accumulation point of the sequence $(\lambda^n)_{n=1}^\infty$. It must therefore be the case that

$$\limsup_{n\to\infty} \lambda^n = 0$$

whence $\lambda^n \to 0$, as required. ■

It follows immediately from these results that $\bar{v}^n \to v^*$ in L^2 and in probability. Observe that

$$\begin{aligned}\mathbb{E}\left[(\bar{v}^n - v^*)^2\right] &= \mathbb{E}\left[(\bar{v}^n - \mathbb{E}\bar{v}^n + \mathbb{E}\bar{v}^n - v^*)^2\right] \\ &= \mathrm{Var}(\bar{v}^n) + (\mathbb{E}\bar{v}^n - v^*)^2 \\ &= \lambda^n\sigma^2 + \left(\delta^n c - \frac{c}{1-\gamma}\right)^2. \end{aligned} \tag{16}$$

Together, Proposition 5 and Theorem 7 imply that (16) vanishes to zero as $n \to \infty$.

Proposition 4 along with Theorem 7 have important practical as well as theoretical implications. The lower bound provided by Proposition 4 ensures that the stepsize will not decline too quickly. While this bound is provided by many standard rules, including $1/n$, we now have the benefit of a rule that is designed to minimize prediction error for faster convergence, but is still guaranteed to avoid the risk of stalling. The guarantee in Theorem 7 that the stepsize will asymptotically approach zero is particularly valuable in applications where we are interested not just in the policy, but in the values themselves. For example, in finance, the value function is used to estimate the price of an option. In the fleet management application of [9], the value functions were used to estimate the marginal value of truck drivers. In both applications, it is essential to have an algorithm that will produce tight estimates of these values.

C. Algorithmic Procedure for General Dynamic Programs

We now discuss how (11) can be adapted for a general dynamic program. The first step is to consider an extension of the single-state model where c and σ^2 are unknown. In this case, we estimate the unknown quantities by smoothing on the observations $\hat{c}^n$ and plugging these estimates into the expression for the optimal stepsize (this is known as the plug-in principle; see, e.g., [44]). Let

$$\bar{c}^n = (1-\nu_{n-1})\bar{c}^{n-1} + \nu_{n-1}\hat{c}^n \tag{17}$$

$$(\bar{\sigma}^n)^2 = (1-\nu_{n-1})\left(\bar{\sigma}^{n-1}\right)^2 + \nu_{n-1}\left(\hat{c}^n - \bar{c}^{n-1}\right)^2 \tag{18}$$

represent our estimates of the mean and variance of the rewards. The secondary stepsize ν_{n-1} is chosen according to some deterministic stepsize rule (e.g., set to a constant). Then, (11) becomes

$$\alpha_{n-1} = \frac{(1-\gamma)\lambda^{n-1}\left(\bar{\sigma}^n\right)^2 + \left(1-(1-\gamma)\delta^{n-1}\right)^2\left(\bar{c}^n\right)^2}{(1-\gamma)^2\lambda^{n-1}\left(\bar{\sigma}^n\right)^2 + \left(1-(1-\gamma)\delta^{n-1}\right)^2\left(\bar{c}^n\right)^2 + \left(\bar{\sigma}^n\right)^2} \tag{19}$$

where δ^{n-1}, λ^{n-1} are computed the same way as before.

At first glance, it appears that we run into the problem of needing a secondary stepsize to calculate an optimal one. However, it is important to note that the secondary stepsize ν_{n-1} is only required to estimate the parameters of the distribution of the one-period reward $\hat{c}^n$. Unlike the sequence $\bar{v}^n$ of value function approximations, the one-period reward in the single-state problem is *stationary*, and can be straightforwardly estimated from the random rewards collected in each time period.

1: Initialize $\bar{V}^0(S,x)$ and $\alpha_0(S,x)$ for all (S,x). Set $\delta^1(S,x)=\alpha_0(S,x)$ and $\lambda^1(S,x)=\alpha_0(S,x)^2$. Also initialize $\bar{c}^0$, $\bar{\sigma}^0$, S^0, and x^0.

2: Set $n=1$, and generate S^1 from the transition function.

3: Solve

$$\hat{v}^n=\max_{x\in\mathcal{X}} C\left(S^n,x\right)+\gamma\bar{V}^{n-1}\left(S^n,x\right)$$

and let x^n be the value of x that achieves the maximum.

4: Update the system-wide parameters

$$\begin{aligned}\bar{c}^n &= \left(1-\nu_{n-1}\right)\bar{c}^{n-1}+\nu_{n-1}C\left(S^n,x^n\right),\\ \left(\bar{\sigma}^n\right)^2 &= \left(1-\nu_{n-1}\right)\left(\bar{\sigma}^{n-1}\right)^2+\nu_{n-1}\left(C\left(S^n,x^n\right)-\bar{c}^{n-1}\right)^2.\end{aligned}$$

5: If $n>1$, calculate

$$\alpha_{n-1}\left(S^{n-1},x^{n-1}\right)=\frac{\left(1-\gamma\right)\lambda^{n-1}\left(S^{n-1},x^{n-1}\right)\left(\bar{\sigma}^n\right)^2+\left(1-\left(1-\gamma\right)\delta^{n-1}\left(S^{n-1},x^{n-1}\right)\right)^2\left(\bar{c}^n\right)^2}{\left(1-\gamma\right)^2\lambda^n\left(S^{n-1},x^{n-1}\right)\left(\bar{\sigma}^n\right)^2+\left(1-\left(1-\gamma\right)\delta^{n-1}\left(S^{n-1},x^{n-1}\right)\right)^2\left(\bar{c}^n\right)^2+\left(\bar{\sigma}^n\right)^2}. \quad (20)$$

6: Update the value function approximation using

$$\bar{V}^n\left(S^{n-1},x^{n-1}\right)=\left(1-\alpha_{n-1}\right)\bar{V}^{n-1}\left(S^{n-1},x^{n-1}\right)+\alpha_{n-1}\hat{v}^n.$$

7: Update the stepsize parameters using

$$\begin{aligned}\delta^n\left(S^{n-1},x^{n-1}\right)&=\alpha_{n-1}\left(S^{n-1},x^{n-1}\right)+\left(1-\left(1-\gamma\right)\alpha_{n-1}\left(S^{n-1},x^{n-1}\right)\right)\delta^{n-1}\left(S^{n-1},x^{n-1}\right),\\ \lambda^n\left(S^{n-1},x^{n-1}\right)&=\alpha_{n-1}^2\left(S^{n-1},x^{n-1}\right)+\left(1-\left(1-\gamma\right)\alpha_{n-1}\left(S^{n-1},x^{n-1}\right)\right)^2\lambda^{n-1}\left(S^{n-1},x^{n-1}\right).\end{aligned}$$

8: Generate S^{n+1} from the transition function or by following a target policy.

9: Increment n and return to Step 3.

Fig. 3. Example implementation of infinite-horizon OSAVI in a finite-state, finite-action MDP with a generic ADP algorithm.

The true significance of (19) is that it can be easily extended to a general MDP with many states and actions. In this case, we replace the random reward $\hat{c}^n$ in (17)–(18) by the one-period reward $C(S^n,x^n)$ earned by taking action x^n in state S^n. The sequence of these rewards depends on the policy used to visit states; in approximate value iteration, this policy will change over time, thus making the sequence of rewards non-stationary. However, as discussed in Section II-A, the basic value-iteration update of (2) eventually converges to an optimal policy, meaning that the expected one-period reward earned in a state converges to a single system-wide constant $\bar{c}$. This suggests that, in a general DP, it is sufficient to keep one single system-wide estimate $\bar{c}^n$ (and similarly $\bar{\sigma}^n$) rather than to store state-dependent estimates.

On the other hand, the quantities δ^n and λ^n are related to the bias and variance of the value function approximation. This suggests that, in a general DP, they should be state-dependent. For example, if we use the Q-learning algorithm, we will have a separate approximation for each state-action pair, leading to a state-dependent stepsize. Fig. 3 describes an example implementation of OSAVI in a classic finite-state, finite-action MDP where a generic ADP algorithm is used with a lookup table approximation. In a more complex problem, if we employ a state aggregation method such as that of [45], we would store a different δ^n and λ^n for each block of the aggregation structure. The memory cost is similar to the procedure in [32], where two recursively updated quantities are stored for each estimated parameter.

In a general ADP setting, we suggest using a constant stepsize in (17), e.g., $\nu_{n-1}=0.2$, to avoid giving equal weight to early observations taken in the transient period before the MDP has reached steady state, while the probability of being in a state is still changing with the policy. Our numerical work suggests that performance is not very sensitive to the choice of ν_{n-1}.

Finally, we briefly note that our convergence analysis in Section III-B mostly carries over to the general case. First, the bound in Proposition 4 still holds almost surely, since the proof holds for any values of c and σ, even if they change between iterations. The bounds in Proposition 2 follow from the functional forms of the updates in Proposition 1, and hold for any arbitrary stepsize sequence. Consequently, Theorem 7 still holds a.s. as long as the sample-based approximations $\bar{c}^n,\bar{\sigma}^n$ do not explode to infinity on any subsequence. If these approximations have a type of convergence (e.g., in probability), we will have $\alpha_{n-1}\to 0$ also in that sense.

D. Extension to Finite Horizon

While it is possible to solve finite horizons using the same algorithmic strategy, we observe that optimal stepsizes vary systematically as a function of the number of time periods to the end of horizon. The best stepsize for states at the end of the horizon is very close to $1/n$, because we do not face the need to sum rewards over a horizon. Optimal stepsizes then increase as we move closer to the first time period.

We can capture this behavior using a finite horizon version (with T time stages) of our single-state, single-action problem. In this setting, approximate value iteration is replaced with approximate dynamic programming. Equations (5) and (6) become

$$\hat{v}_t^n=\hat{c}_t^n+\gamma\bar{v}_{t+1}^{n-1} \quad (21)$$

$$\bar{v}_t^n=(1-\alpha_{n-1,t})\bar{v}_t^{n-1}+\alpha_{n-1,t}\hat{v}_t^n. \quad (22)$$

These equations are solved for $t=1,\ldots,T-1$ in each time step n. We assume that $\bar{v}_T^n=0$ for all n, and that the observations $\hat{c}_t^n$ are independent and identically distributed for all n and t.

Our analysis can easily be extended to this setting. First, we can obtain expressions for the expected value and variance of $\bar{v}_t^n$ that generalize our derivations of δ^n and λ^n in Section III-A. The following proposition describes these expressions.

Proposition 6: For $t=1,\ldots,T-1$, define

$$\delta_t^n=\begin{cases}\alpha_{0,t} & n=1\\ \left(1+\gamma\delta_{t+1}^{n-1}\right)\alpha_{n-1,t}+(1-\alpha_{n-1,t})\delta_t^{n-1} & n>1.\end{cases}$$

Also, for $t, t' = 1, \ldots, T-1$, let

$$\lambda^n_{t,t'} = \begin{cases} \alpha^2_{0,t} 1_{\{t=t'\}} & n = 1 \\ \alpha^2_{n-1,t} 1_{\{t=t'\}} + J^{n-1}_{t,t'} + K^{n-1}_{t,t'} + L^{n-1}_{t,t'} + M^{n-1}_{t,t'} & n > 1 \end{cases}$$

where

$$\begin{aligned} J^{n-1}_{t,t'} &= (1-\alpha_{n-1,t})(1-\alpha_{n-1,t'})\lambda^{n-1}_{t,t'}, \\ K^{n-1}_{t,t'} &= \gamma(1-\alpha_{n-1,t})\alpha_{n-1,t'}\lambda^{n-1}_{t,t'+1}, \\ L^{n-1}_{t,t'} &= \gamma\alpha_{n-1,t}(1-\alpha_{n-1,t'})\lambda^{n-1}_{t+1,t'}, \\ M^{n-1}_{t,t'} &= \gamma^2\alpha_{n-1,t}\alpha_{n-1,t'}\lambda^{n-1}_{t+1,t'+1}. \end{aligned}$$

Then, $\mathbb{E}(\bar{v}^n_t) = \delta^n_t c$ and $Cov(\bar{v}^n_t, \bar{v}^n_{t'}) = \lambda^n_{t,t'}\sigma^2$.

The proof uses the same logic as the proof of Proposition 1. We can think of λ^n as a symmetric matrix that can be updated recursively using the elements of λ^{n-1}. The matrix starts out diagonal, and as n increases, the covariances gradually expand from the main diagonal outward. Next, we can repeat the analysis of Section III-A to solve

$$\min_{\alpha_{n-1,t}\in[0,1]} \mathbb{E}\left[(\bar{v}^n_t(\alpha_{n-1,t}) - \mathbb{E}\hat{v}^n_t)^2\right].$$

The next result gives the solution.

Theorem 8: If $\alpha_{0,t}$ is given, the optimal stepsize for time t at iteration n is given by (23), as shown at the bottom of the page.

In the infinite-horizon case, this reduces to our original formula in (11). The finite-horizon formula requires us to store more parameters in the form of a matrix λ^n, which has the potential to incur substantially greater computational cost. The benefit is that we can now optimally vary the stepsize by t. If c and σ^2 are unknown, we can adapt the approximation procedure outlined in Section III-C, and replace the unknown values in (23) with $\bar{c}^n$ and $(\bar{\sigma}^n)^2$.

IV. Experimental Study: One State, One Action

We first study the performance of our stepsize rule on an instance of the single-state, single-action problem. This allows us to obtain insights into the sensitivity of performance with respect to different problem parameters. We considered normally distributed rewards with mean $c = 1$ and standard deviation $\sigma = 1$, with $\gamma = 0.9$ as the discount factor. The optimal value for this problem is $V^* = 10$. All policies used $\bar{v}^0 = 0$ as the initial approximation. Furthermore, all sample-based parameters for these policies (e.g., $\bar{c}^0$ and $\bar{\sigma}^0$ for OSAVI) were initialized to zero here and throughout all parts of our study. Five different stepsize rules were implemented; we briefly describe them as follows.

Optimal Stepsize for Approximate Value Iteration (OSAVI): We use the approximate version of the optimal stepsize, given by (20). The secondary stepsize ν_{n-1} was set to 0.2.

Bias-Adjusted Kalman Filter (OSA/BAKF): We use the approximate version of the OSA/BAKF algorithm in [37, Fig. 4]. Like OSAVI, this stepsize minimizes a form of the prediction error for a scalar signal processing problem, but assumes that observations are independent. A secondary stepsize rule $\bar{\nu}_{n-1} = 0.05$ is used to estimate the bias of the value function approximation (unlike OSAVI, which uses a closed-form expression for this quantity).

McClain's Rule: McClain's stepsize formula is given by

$$\alpha_n = \begin{cases} 1 & \text{if } n = 1 \\ \frac{\alpha_{n-1}}{1+\alpha_{n-1}-\bar{\alpha}} & \text{otherwise} \end{cases}$$

where $\bar{\alpha}$ is a tunable parameter. This stepsize behaves like the $1/n$ rule in early iterations, but quickly converges to the limit point $\bar{\alpha}$, and then behaves more like a constant stepsize rule. This tends to happen within approximately 10 iterations. For our experiments, we used $\bar{\alpha} = 0.1$; the issue of tuning $\bar{\alpha}$ is discussed in Section IV-B. McClain's rule should be viewed as a slightly more sophisticated version of a constant stepsize.

Harmonic Stepsize: This deterministic rule is given by $\alpha_{n-1} = a/(a+n)$, where $a > 0$ is a tunable parameter. A value of $a = 10$ yielded good performance for our choice of problem parameters. However, the harmonic stepsize is sensitive to the choice of the tunable parameter a, which is highly problem dependent. If we expect good convergence in a few hundred iterations, a on the order of 5 or 10 may work quite well. On the other hand, if we anticipate running our algorithm millions of iterations (which is not uncommon in reinforcement learning), we might choose a on the order of 10,000 or higher. This issue is discussed further in Section IV-B.

Incremental Delta-Bar-Delta (IDBD): This rule, introduced by [33], is given by $\alpha_{n-1} = \min\{1, \exp(\Delta_{n-1})\}$, where $\Delta_n = \Delta_{n-1} + \theta(\hat{v}^n - \bar{v}^{n-1})h_{n-1}$ and $h_n = (1-\alpha_{n-1})h_{n-1} + \alpha_{n-1} \times (\hat{v}^n - \bar{v}^{n-1})$. This is an example of an exponentiated gradient method, where averaging is performed on the logarithm of the stepsize. We used $\theta = 0.2$ as the tunable parameter.

We also considered the polynomial stepsize $\alpha_{n-1} = 1/n^\beta$, but it consistently underperformed the rules listed above, and is omitted from the subsequent analysis. The constant rule $\alpha_{n-1} = \bar{\alpha}$ yielded results very similar to McClain's rule, and is also omitted.

A. Numerical Evaluation of Stepsize Rules

Fig. 4 shows the value of the objective function in (8) achieved by each stepsize rule over 10^4 iterations. The OSAVI rule consistently achieves the best performance (lowest objective value). However, the harmonic stepsize, when properly tuned, performs comparably. The BAKF and McClain rules level off around an objective value of 10^{-2}. Each data point in Fig. 4 is an average over an "outer loop" of 10^4 simulations.

$$\alpha_{n-1,t} = \frac{\left(\lambda^{n-1}_{t,t} - \gamma\lambda^{n-1}_{t,t+1}\right)\sigma^2 + \left(1 - \delta^{n-1}_t + \gamma\delta^{n-1}_{t+1}\right)^2 c^2}{\left(\lambda^{n-1}_{t,t} - 2\gamma\lambda^{n-1}_{t,t+1} + \gamma^2\lambda^{n-1}_{t+1,t+1}\right)\sigma^2 + \left(1 - \delta^{n-1}_t + \gamma\delta^{n-1}_{t+1}\right)^2 c^2 + \sigma^2}. \tag{23}$$

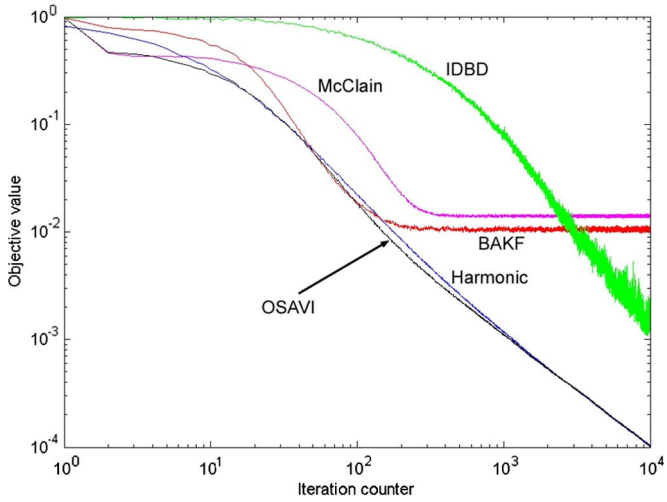


Fig. 4. Objective values achieved by each stepsize rule over 10^4 iterations.

It should be noted that, while IDBD displays the slowest convergence early on, it eventually overtakes BAKF and McClain's rule and continues to exhibit improvement in the late iterations. In the single-state setting, we found that it was less sensitive to its tunable parameter than the harmonic rule, and also produced less volatile stepsizes than BAKF. We will examine the performance of this rule in multi-stage problems later on. By contrast, the other benchmarks (harmonic, McClain, and BAKF) were fairly sensitive to their tunable parameters. We discuss this below in the context of the single-state problem, which allows us to examine tuning issues with a minimal number of other problem inputs.

B. Discussion of Tunable Parameters

We begin by considering the approximate BAKF rule, which uses a secondary stepsize $\bar{\nu}_{n-1}$ to estimate the bias. Fig. 5(a) shows the effect of varying $\bar{\nu}_{n-1}$ on the objective value achieved by BAKF (with the optimal stepsize shown for comparison). We see that, when we use a constant value for the secondary stepsize (e.g., $\bar{\nu}_{n-1} = 0.05$), there is a clear tradeoff between performance in the early and late iterations. Smaller values of $\bar{\nu}_{n-1}$ result in better performance in the long run (the objective value achieved by BAKF plateaus at a lower level), but worse performance in the short run. In terms of the quality of our approximation of V^*, smaller constants cause slower convergence, but more stable estimates.

It is also necessary to note one special case, where $\bar{\nu}_{n-1} = 1/n$. If we use the $1/n$ rule for the secondary stepsize, the objective value achieved by BAKF declines to zero in the long run. This is not the case when we use a constant stepsize. Furthermore, the use of the $1/n$ rule produces very close performance to that of OSAVI. However, in a general MDP, where there are many different rewards, a constant stepsize may be better able to handle the transient phase of the MDP. For this reason, we focus primarily on constant values of $\bar{\nu}_{n-1}$ in this study.

Even with a declining secondary stepsize, the BAKF rule is outperformed by OSAVI with a simple constant secondary stepsize of $\nu_{n-1} = 0.2$. The results for different values of $\bar{\nu}_{n-1}$ indicate that BAKF is quite sensitive to the choice of $\bar{\nu}_{n-1}$.

Fig. 5(b) suggests that OSAVI is relatively insensitive to the choice of secondary stepsize. The lines in Fig. 5(b) represent the performance of OSAVI for values of ν_{n-1} ranging from 0.05 to as high as 0.5. We see that these changes have a much smaller effect on the performance of OSAVI than varying $\bar{\nu}_{n-1}$ had on the BAKF rule. Very small values of ν_{n-1}, such as 0.05, do yield slightly poorer performance, but there is little difference between 0.2 and 0.5. Furthermore, the objective value achieved by OSAVI declines to zero for each constant value of ν_{n-1}, whereas BAKF always levels off under a constant secondary stepsize. We conclude that OSAVI is more robust than BAKF, and requires less tuning of the secondary stepsize.

Fig. 6(a) shows the sensitivity of McClain's rule to the choice of tunable parameter $\bar{\alpha}$. The effect is very similar to the effect of using different constant values of $\bar{\nu}_{n-1}$ in Fig. 5(a). Smaller values of $\bar{\alpha}$ give better (more stable) late-horizon performance and worse (slower) early-horizon performance.

The harmonic rule is analyzed in Fig. 6(b). We see that $a = 10$ is a good choice for this problem, with the particular parameter values (variance and discount factor) that we have chosen. Larger values of a are consistently worse, and smaller values are only effective in the very early iterations. However, $a = 10$ yields very good performance, the best out of all the competing stepsize rules.

In fact, it is possible to tune the harmonic rule to perform competitively against OSAVI. However, the best value of a is highly problem-dependent. Fig. 7(a) shows that $a = 10$ continues to perform well when σ^2 is increased to 4, and even achieves a slightly lower objective value than the approximate optimal rule in the later iterations, although OSAVI performs noticeably better in the early iterations. However, Fig. 7(b) shows that $a = 100$ becomes the best value when the discount factor γ is increased to 0.99. The optimal rule has not been retuned in Fig. 7; all results shown are for $\nu_{n-1} = 0.2$. Interestingly, it appears that the optimal choice of a is more sensitive to the discount factor than to the signal-to-noise ratio.

We conclude based on Figs. 6(b) and 7 that the best choice of a in the harmonic stepsize rule is very sensitive to the parameters of the problem, and that the best choice of a for one problem setting can perform very poorly for a different problem. By contrast, Fig. 5(b) shows that OSAVI is relatively insensitive to its tunable parameter. A simple value of $\nu_{n-1} = 0.2$ yields good results in all of the settings considered. We claim that OSAVI is a robust alternative to several leading stepsize rules.

V. Experimental Study: General MDP

We also tested the general OSAVI rule from Section III-C on a synthetic MDP with 100 states and 10 actions per state, generated in the following manner. For state S and action x, with probability 0.8 the reward $C(S, x)$ was generated uniformly on [0, 2], and with probability 0.2 it was generated uniformly on [18, 20]. For each (S, x), we randomly picked 10 states to be reachable. For each such state S', we generated a number $b_{S,S',x} \sim U[0, 1]$ and let $b_{S,S',x} / \sum_{S''} b_{S,S'',x}$ be the probability of making a transition to S' out of (S, x). The transition probability to any state not reachable from (S, x) was

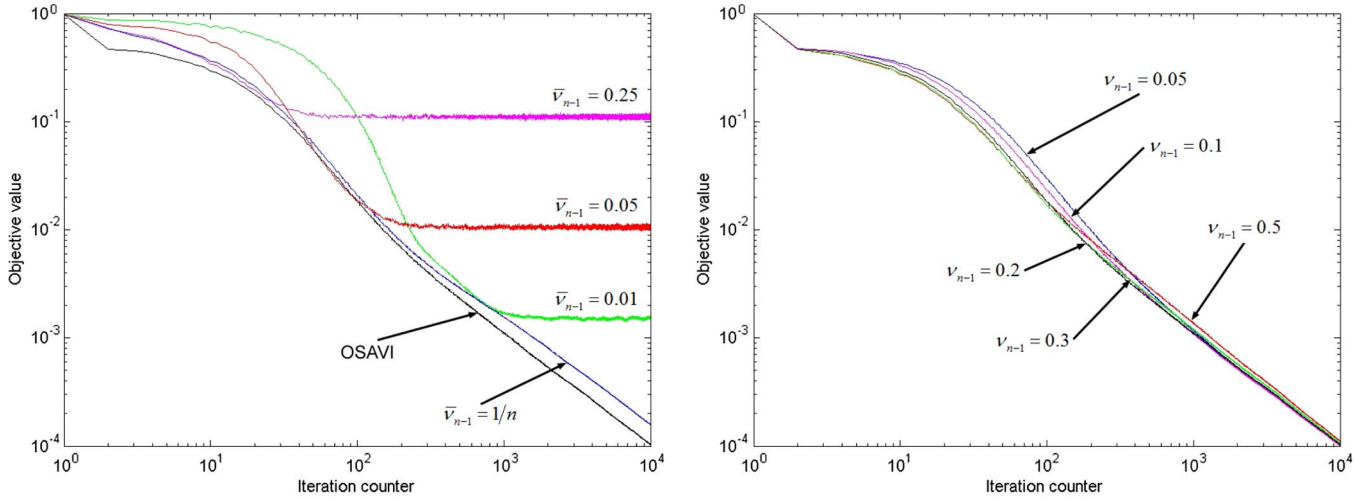


Fig. 5. Effect of the secondary parameter ν_{n-1} on the objective values achieved by (a) the approximate BAKF rule, and (b) the approximate optimal rule. (a) BAKF rule. (b) OSAVI rule.

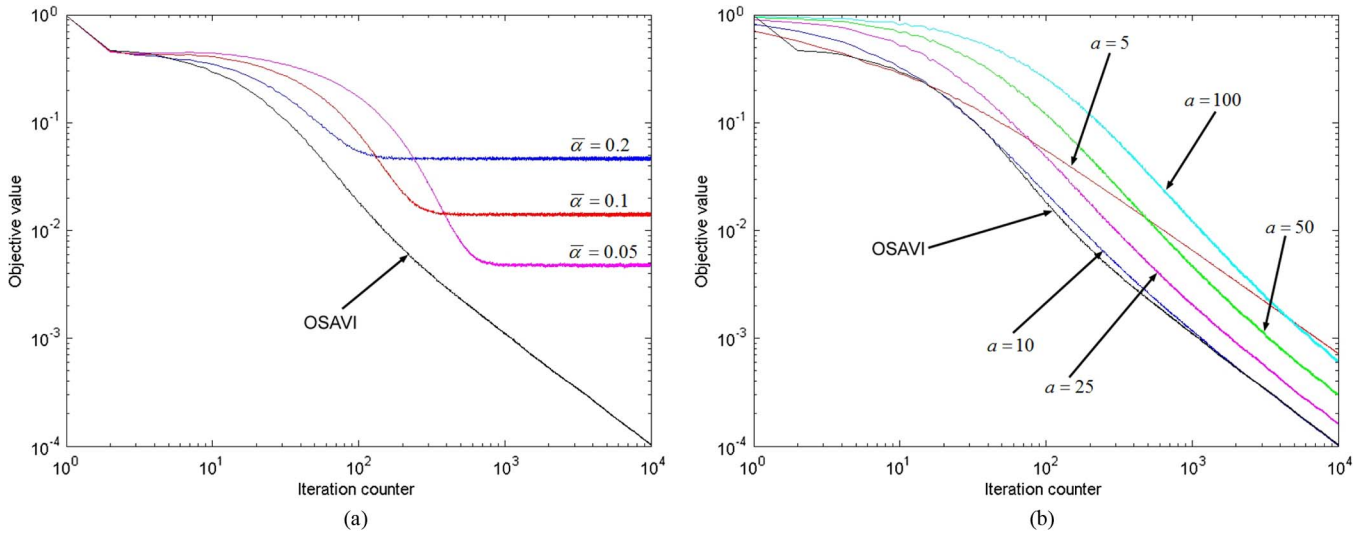


Fig. 6. Sensitivity of (a) McClain's rule and (b) the harmonic rule to their respective tunable parameters.

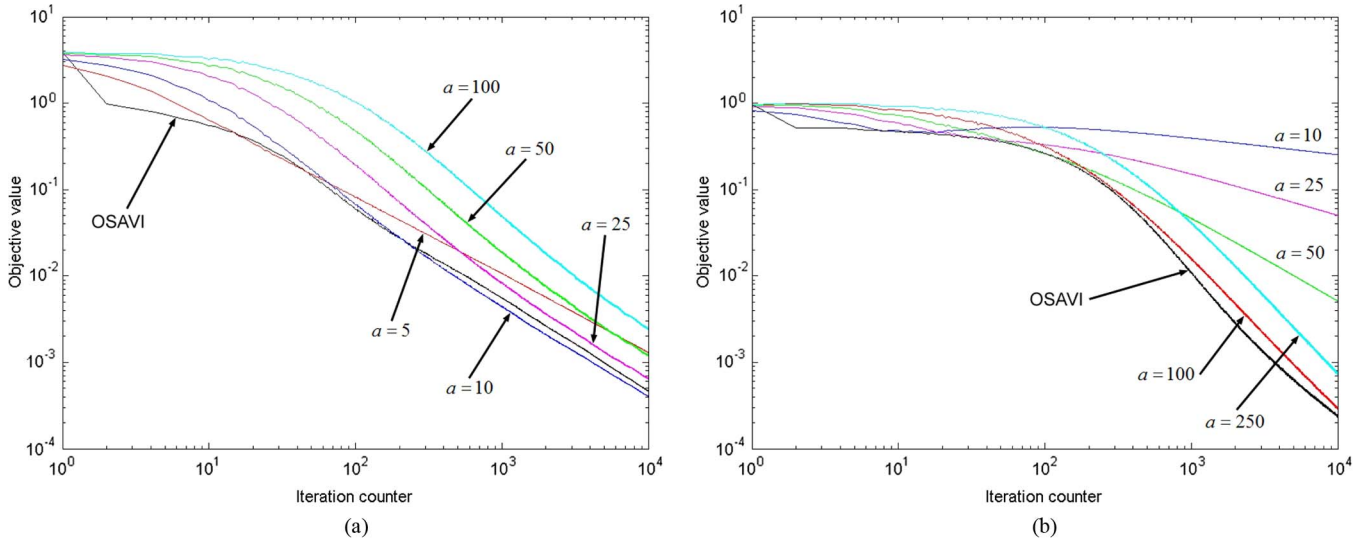


Fig. 7. Sensitivity of the harmonic stepsize rule in different problem settings. (a) $\sigma^2 = 4$; (b) $\gamma = 0.99$.

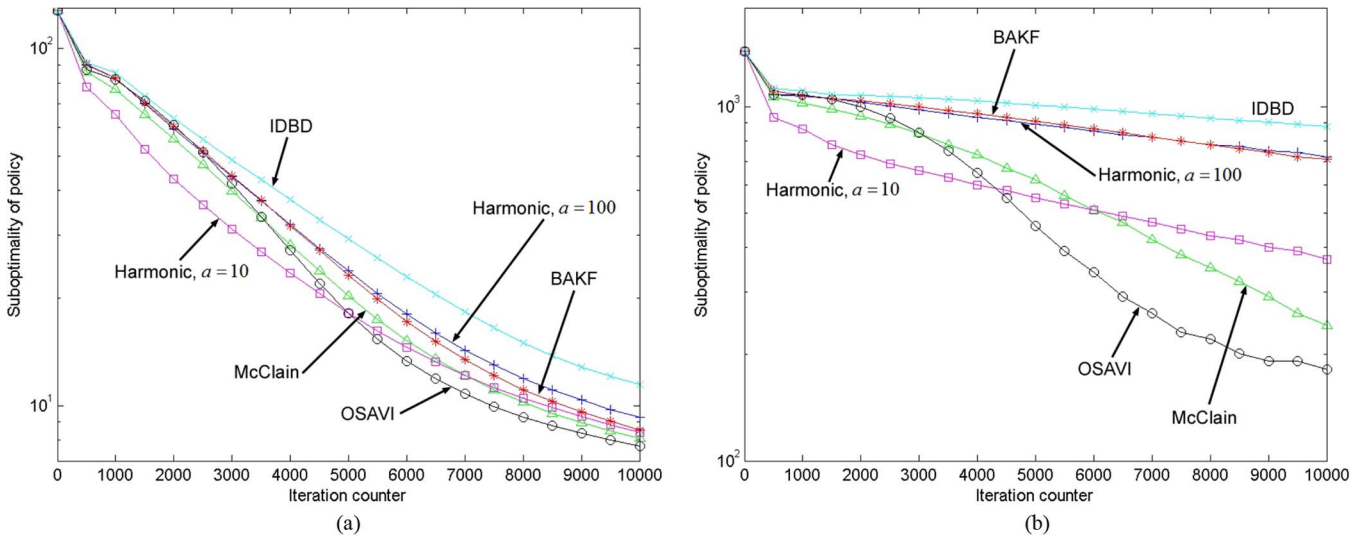


Fig. 8. Suboptimality for each stepsize rule for $\gamma = 0.9$ and $\gamma = 0.99$ in the 100-state MDP; (a) $\gamma = 0.9$; (b) $\gamma = 0.99$.

zero. In this manner, we obtained a sparse MDP with some high-value states, leading to some variety in the value function. We used value iteration to compute the true optimal value for $\gamma = 0.9$ and $\gamma = 0.99$.

A. *Infinite-Horizon Setting*

Each stepsize was implemented together with the following off-policy approximate value iteration algorithm. The value function approximation $\bar{V}^0(S, x) = 0$ is defined for each state-action pair (S, x), as in the Q-learning algorithm. Upon visiting state S^n, an action x^n is chosen uniformly at random. Then, a new state S' is simulated from the transition probabilities of the MDP. We then compute

$$\hat{v}^n = \max_x C(S', x) + \gamma \bar{V}^{n-1}(S', x),$$
$$\bar{V}^n(S^n, x^n) = (1 - \alpha_{n-1})\bar{V}^{n-1}(S^n, x^n) + \alpha_{n-1}\hat{v}^n$$

where α_{n-1} is chosen according to some stepsize rule. The next state S^{n+1} to be visited in the next iteration is then chosen uniformly at random (not set equal to S').

We briefly discuss the reasoning behind this design. Any policy that uses the value function approximation to make decisions will also implicitly depend on the stepsize used to update that approximation. The stepsize affects the policy, which then affects the sequence (and frequency) of visited states, which in turn affects the calculation of future stepsizes. Ensuring that "good" states are visited sufficiently often is very important to the practical performance of ADP algorithms, but this issue (known as the problem of exploration) is quite separate from the problem of stepsize selection, and is outside the scope of our paper. We have sought to decouple the stepsize from the ADP policy by randomly generating states and actions.

We ran the above algorithm for 10^4 iterations with each of the five stepsize rules from Section IV. Performance after N iterations can be evaluated as follows. We find the policy π that takes the action $\arg\max_x C(S, x) + \gamma \bar{V}^N(S, x)$ at state S, and then calculate the value $V^\pi = (I - \gamma P^\pi)^{-1} C^\pi$, where $P^\pi_{S,S'}$ is the probability of transitioning from S to S' under the policy π, and $C^\pi(S)$ is the reward obtained by following π in state S. Then, we calculate

$$\frac{1}{|\mathcal{S}|}\sum_S V^*(S) - V^\pi(S)$$

where V^* is the true value function obtained from value iteration. This gives us the suboptimality of the π. We average this quantity over 10^4 simulations, each consisting of N iterations of the learning algorithm. With 10^4 simulations, the standard errors of this performance measure are negligible relative to its magnitude, and are omitted from the subsequent figures and discussion.

We compared the following stepsize rules: McClain's rule with $\bar{\alpha} = 0.1$, the harmonic rule with $a = 10$ and $a = 100$ (these are the tuned values that were found in Section IV-B to work best for $\gamma = 0.9$ and $\gamma = 0.99$, respectively), the BAKF rule of [37] with a secondary stepsize of 0.05, and OSAVI with a secondary stepsize of 0.2. For IDBD, we experimented with several orders of magnitudes for θ, and found that $\theta = 0.001$ produced good performance, although the difference between magnitudes was relatively small. We also made the stepsizes state-dependent in order to achieve quicker convergence. For example, the harmonic rule is given by $\alpha_{n-1}(S) = a/(a + N^n(S))$ where $N^n(S)$ is the number of times state S was visited in n iterations. The parameters δ^n, λ^n used by OSAVI, and similar parameters for BAKF, were chosen to be state-dependent.

Fig. 8 shows the average suboptimality achieved by each stepsize rule over time, up to 10^4 iterations. We see that, for both discount factors, the harmonic rule with $a = 10$ achieves the best performance early on, but slows down considerably in later iterations. OSAVI achieves the best performance in the second half of the time horizon, and the margin of victory is more clearly pronounced for $\gamma = 0.99$.

We conclude that OSAVI yields generally competitive performance. The harmonic rule can be tuned to perform well, but performance is quite sensitive to the value of a, which is particularly visible in Fig. 8(b). The secondary parameter for OSAVI was not tuned at all, as we wish to observe that a single constant value is sufficient to produce competitive performance.

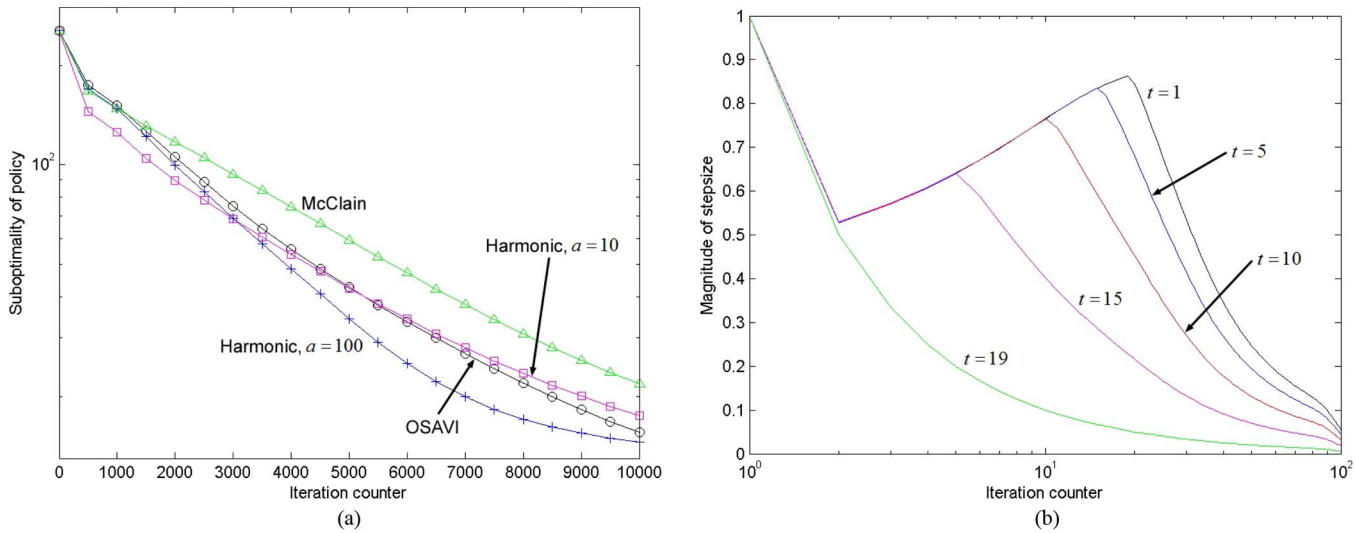


Fig. 9. Finite-horizon results: (a) Suboptimality for different stepsize rules. (b) Magnitudes of $\alpha_{n-1,t}$ for finite-horizon OSAVI.

B. Finite-Horizon Setting

Finite-horizon problems introduce the dimension that the relative size of the learning bias versus the noise in the contribution depends on the time period. As a result, the optimal stepsize behavior changes with time. In the finite-horizon case, we run the same off-policy algorithm as before, with the update now calculated via the equations

$$\hat{v}_t^n = \max_x C\left(S_t', x\right) + \gamma \bar{V}_{t+1}^{n-1}(S', x)$$
$$\bar{V}_t^n\left(S_t^n, x_t^n\right) = (1 - \alpha_{n-1,t})\bar{V}_t^{n-1}\left(S_t^n, x_t^n\right) + \alpha_{n-1}\hat{v}_t^n$$

for $t = 1, \ldots, T-1$. The true value $V_t(S)$ of being in state S at time t can be found using backward dynamic programming; in the following, we use the values at $t = 1$ to evaluate all policies. Our performance measure is again the suboptimality of the policy induced by the value function approximation, averaged over all states. We used the same MDP as in Section V-A with the horizon $T = 20$, and the discount factor $\gamma = 0.99$.

We compared the approximate version of the finite-horizon OSAVI rule from (23) to McClain's rule with $\bar{\alpha} = 0.1$ and the harmonic rule $\alpha_{n-1,t} = a/(a+n)$ with $a = 10$ and $a = 100$. These rules achieved the best performance in the previous experiments, and can be easily applied to a finite-horizon problem. As before, all stepsizes were made to be state-dependent. Fig. 9(a) shows the average suboptimality of each stepsize rule. We see that the harmonic rule is competitive with OSAVI overall. However, the performance of $a = 10$ slows down in later iterations, as in Fig. 8. Furthermore, while $a = 100$ outperforms OSAVI in the mid- to late iterations, OSAVI has largely closed the gap by the end and continues to improve, while the harmonic rule again slows down.

Finally, Fig. 9(b) shows the magnitude of the stepsize $\alpha_{n-1,t}$ produced by the OSAVI formula in a simple synthetic MDP where all 100 states are reachable from each (S, x) and transition probabilities are normalized i.i.d. samples from a $U[0,1]$ distribution. Our purpose here is to illustrate the behavior of the optimal stepsize for different t. When $t = 19$, OSAVI is identical to the $1/n$ rule, as we would expect, since this is the last time in the horizon. We assume $V_{20}(S) = 0$ for all S, so the observations $\hat{v}_{19}^n$ are stationary, and Corollary 6 applies. For values of t earlier in the time horizon, the optimal stepsize steadily increases, with the largest values of $\alpha_{n-1,t}$ being for $t = 1$. It takes a long time for our observations to propagate backward across the time horizon, and so we need larger stepsizes at time $t = 1$ to ensure that these observations have an effect. We note that, for earlier time periods, OSAVI goes through a period of exploration before settling on a curve which can be closely approximated by $a/(a+n)$ for a suitably calibrated choice of a. For the finite-horizon problem, a should be different for each time period.

VI. Experimental Study: ADP for a Continuous Inventory Problem

The last part of our experimental study demonstrates how OSAVI can be used in conjunction with ADP on a problem where the state space is continuous. We present a stylized inventory problem where a generic resource can be bought and sold on the spot market, and held in inventory in the interim. The basic structure of our problem appears in applications in finance [46], energy [5], inventory control [29], and water reservoir management [47]. We deliberately abstract ourselves from any particular setting, as we wish to keep the focus on the stepsize rule and test it in a generic setting for which ADP is required.

The state variable of the generic inventory problem contains two dimensions. Let $S_t = (R_t, P_t)$, where R_t denotes the amount of resource currently held in inventory, and P_t denotes the current spot price of the resource. We assume that R_t can take values in the set $\{0, 0.02, 0.04, \ldots, 1\}$, representing a percentage of the total inventory capacity $\bar{R}$. The action x_t represents our decision to buy more inventory (positive values) or sell from our current stock (negative values). We assume that we can buy or sell up to 50% of the total capacity in one time step, again in increments of 2%. Thus, there are up to 50 actions in the problem. The reward $C(P, x) = -P \cdot \bar{R} \cdot x$ represents the revenue obtained (or cost incurred) after making decision x given a price P.

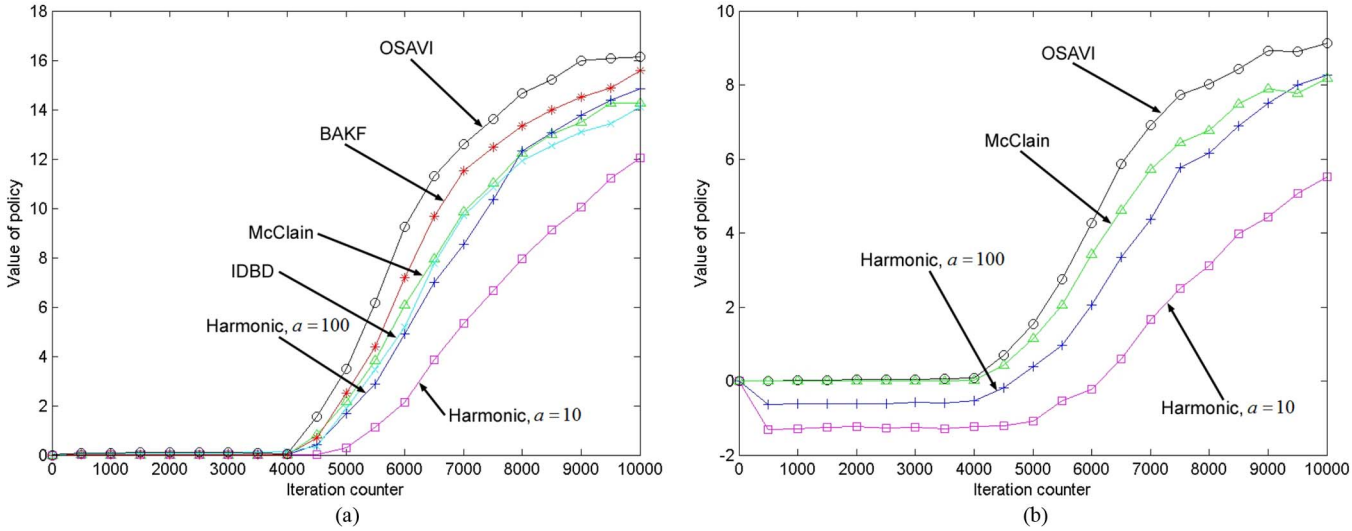


Fig. 10. Offline policy values for different stepsize rules in the inventory problem with (a) infinite horizon and (b) finite horizon.

While the resource variable R_t is discrete, we assume that the spot price P_t is continuous, and follows a geometric Ornstein-Uhlenbeck (mean reverting) process, a standard price model in finance and other areas. With minor modifications to the problem, P_t could also be changed into an exogenous supply process, which we could draw from to satisfy a demand. The important aspect is that P_t is continuous, which makes it impossible to solve (1) for every state. Furthermore, even for a given state S_t, computing the expectation in (1) is difficult, because the transition to the next state S_{t+1} depends on a continuous random variable. For these reasons, we approach the problem using approximate dynamic programming with a discrete value function approximation. To address the issue of the continuous transition to the next state, we use the post-decision state concept introduced in [18] and discussed extensively by [17]. Given a state S_t and a decision x_t, the post-decision state $S_t^x = (R_t^x, P_t^x)$ is given by the equations

$$R_t^x = R_t + x_t,$$
$$P_t^x = P_t.$$

The next pre-decision state S_{t+1} is then obtained by setting $R_{t+1} = R_t^x$ and simulating P_{t+1} from the price process. Given a value function approximation $\bar{V}^{n-1}$, the update $\hat{v}_t^n$ is computed using

$$\hat{v}_t^n = \max_{x_t} C\left(P_t^n, x_t\right) + \gamma \bar{V}_t^{n-1}\left(S_t^{x,n}\right).$$

This quantity is then used to update the previous post-decision state, that is

$$\bar{V}_{t-1}^n\left(S_{t-1}^{x,n}\right) = (1-\alpha_{n-1,t-1})\bar{V}_{t-1}^{n-1}\left(S_{t-1}^{x,n}\right) + \alpha_{n-1,t-1}\hat{v}_t^n.$$

Thus, we can adaptively improve our value function approximation without computing an expectation.

For our value function approximation, we used a lookup table where the log-price $\log P_t$ was discretized into 34 intervals of width 0.125 between -2 and 2. Thus, the table contained a total of $51 \cdot 34 = 1734$ entries, with each entry initialized to a large value of 10^4, in keeping with the recommendation in [17, Sec. 4.9.1] to use optimistic initial estimates. However, while the approximation used a discretized state space, our experiments simulated P_t using the continuous price process. The price was only discretized during calls to the lookup table. This is an important detail: while we use a discrete value function approximation, we are still solving the original continuous problem.

The price process was instantiated with $P_0 = 30$, mean-reversion parameter 0.0633 and volatility 0.2. Most prices are thus around \$30, but sharp spikes are possible. As before, we use a pure exploration policy where each action x_t is chosen uniformly at random. Also as before, we simulate a future state from the price process in order to compute $\hat{v}_t^n$, but the next state to actually be visited by the algorithm is generated randomly (the resource level is generated uniformly at random, and the log-price is generated uniformly between -2.125 and 2.125). Recall that this is necessary in order to separate the performance of the stepsize from the quality and architecture of the value function approximation.

We used the same policies as in Section V: McClain's rule with $\bar{\alpha} = 0.1$, the harmonic rule with $a = 10$, the BAKF rule of [37] with a secondary stepsize of 0.05, IDBD with $\theta = 0.001$, and OSAVI with a secondary stepsize of 0.2. To evaluate the performance of each stepsize rule after N iterations, we fixed the approximation $\bar{V}^N$ and then simulated the total reward obtained by making decisions of the form $x_t = \arg\max_x C(P_t, x) + \gamma V_t^N(S_t^x)$ in both finite- and infinite-horizon settings. This quantity was averaged over 2.5×10^4 sample paths. Fig. 10 reports the performance of the approximation obtained using different stepsize rules. Since our objective is to maximize revenue, higher numbers on the y-axis represent better quality.

Fig. 10(a) shows the performance of OSAVI in the infinite-horizon setting. Because of the larger size of the inventory problem, we require several thousand iterations in order to obtain any improvement in the target policy specified by $\bar{V}^N$. After 4000 iterations, we find that OSAVI consistently yields the most improvement in the value of the target policy. Analogously to Fig. 9(a) in Section V-B, we also compared OSAVI

to the harmonic rule in a finite-horizon setting; the results are shown in Fig. 10(b). As in the infinite-horizon setting, several thousand iterations are required before any improvement can be observed, but OSAVI consistently outperforms the best version of harmonic.

Our experiments on the inventory problem offer additional evidence that our new stepsize rule can be applicable to more complex dynamic programming problems, which cannot be solved exactly, and where additional techniques such as the post-decision state variable are necessary to deal with continuous state spaces and difficult expectations. Even in the streamlined form considered here, the inventory problem features a continuous price variable, and the value of being in a state depends on a mean-reverting stochastic differential equation. The fact that OSAVI retains its advantages over other stepsize rules in this setting is an encouraging sign.

VII. Conclusion

We have derived a new optimal stepsize minimizing the prediction error of the value function approximation in the single-state model. This stepsize is the first to take into account the covariance between the observation we make of the value of being in a state, and our approximation of that value, a property that is inherent in approximate value iteration. Furthermore, we are able to compute a closed-form expression for the prediction bias in the single-state, single-action case, considerably simplifying the task of estimating this quantity in the general case. The rule can be easily extended to a general MDP setting, both finite- and infinite-horizon.

We have tested our stepsize rule against several leading deterministic and stochastic rules. While some competing rules (particularly the harmonic rule) can be tuned to yield very competitive performance, they are also very sensitive to the choice of tuning parameter. On the other hand, our stepsize rule is robust, displaying little sensitivity to the parameter used to estimate the one-period reward. We also tested our stepsize rule on a general discrete-state MDP, as well as on a more complex ADP problem. We found that OSAVI performs competitively against the other rules in both finite- and infinite-horizon settings.

We conclude that our stepsize rule can be a good alternative to other leading stepsizes. Our conclusion reflects the particular set of experiments that we chose to run. It is important to remember that deterministic stepsizes such as the harmonic rule can be finely tuned to a particular problem, resulting in better performance than the adaptive rule that we present. The strength of our rule, however, is its ability to adjust to the evolution of the value function approximation, as well as its relative lack of sensitivity to tuning.

Acknowledgment

The authors wish to thank the members of the Topology Atlas Forum for several very helpful discussions, and C. Szepesvári and three referees whose comments led to significant improvements in the paper.

References

[1] D. Adelman and D. Klabjan, "Computing near-optimal policies in generalized joint replenishment," *INFORMS J. Comp.*, vol. 24, no. 1, pp. 148–164, 2012.

[2] M. S. Maxwell, M. Restrepo, S. G. Henderson, and H. Topaloglu, "Approximate dynamic programming for ambulance redeployment," *INFORMS J. Comp.*, vol. 22, no. 2, pp. 266–281, 2010.

[3] M. He, L. Zhao, and W. B. Powell, "Approximate dynamic programming algorithms for optimal dosage decisions in controlled ovarian hyperstimulation," *Eur. J. Oper. Res.*, vol. 222, no. 2, pp. 328–340, 2012.

[4] G. Lai, F. Margot, and N. Secomandi, "An approximate dynamic programming approach to benchmark practice-based heuristics for natural gas storage valuation," *Oper. Res.*, vol. 58, no. 3, pp. 564–582, 2010.

[5] N. Löhndorf and S. Minner, "Optimal day-ahead trading and storage of renewable energies - an approximate dynamic programming approach," *Energy Syst.*, vol. 1, no. 1, pp. 61–77, 2010.

[6] N. Secomandi, "Optimal commodity trading with a capacitated storage asset," *Manag. Sci.*, vol. 56, no. 3, pp. 449–467, 2010.

[7] D. Zhang and D. Adelman, "An approximate dynamic programming approach to network revenue management with customer choice," *Transportation Sci.*, vol. 43, no. 3, pp. 381–394, 2009.

[8] D. A. Castanon, "Approximate dynamic programming for sensor management," in *Proc. 36th IEEE Conf. Decision Control*, 1997, vol. 2, pp. 1202–1207.

[9] H. P. Simão, J. Day, A. P. George, T. Gifford, J. Nienow, and W. B. Powell, "An approximate dynamic programming algorithm for large-scale fleet management: A case application," *Transportation Sci.*, vol. 43, no. 2, pp. 178–197, 2009.

[10] W. B. Powell, A. George, A. Lamont, J. Stewart, and W. R. Scott, "SMART: A stochastic multiscale model for the analysis of energy resources, technology and policy," *INFORMS J. Comp.*, vol. 24, no. 4, pp. 665–682, 2012.

[11] R. Howard, *Dynamic Probabilistic Systems, Volume II: Semimarkov and Decision Processes*. New York, NY, USA: Wiley, 1971.

[12] M. L. Puterman, *Markov Decision Processes*. New York, NY, USA: Wiley, 1994.

[13] R. Bellman and S. Dreyfus, "Functional approximations and dynamic programming," *Math. Tables Aids Comp.*, vol. 13, pp. 247–251, 1959.

[14] D. Bertsekas and J. Tsitsiklis, *Neuro-Dynamic Programming*. Belmont, MA, USA: Athena Scientific, 1996.

[15] R. Sutton and A. Barto, *Reinforcement Learning*. Cambridge, MA, USA: The MIT Press, 1998.

[16] J. Si, A. G. Barto, W. B. Powell, and D. Wunsch, Eds., *Handbook of Learning and Approximate Dynamic Programming*. New York, NY, USA: IEEE Press, 2004.

[17] W. B. Powell, *Approximate Dynamic Programming: Solving the Curses of Dimensionality (2nd ed.)*. New York, NY, USA: Wiley, 2011.

[18] B. Van Roy, D. Bertsekas, Y. Lee, and J. Tsitsiklis, "A neuro-dynamic programming approach to retailer inventory management," in *Proc. 36th IEEE Conf. Decision Control*, 1997, vol. 4, pp. 4052–4057.

[19] C. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3, pp. 279–292, 1992.

[20] H. Topaloglu and W. B. Powell, "Dynamic programming approximations for stochastic, time-staged integer multicommodity flow problems," *INFORMS J. Comp.*, vol. 18, no. 1, pp. 31–42, 2006.

[21] M. Wasan, *Stochastic Approximation*. Cambridge, U.K.: Cambridge Univ. Press, 1969.

[22] H. J. Kushner and G. G. Yin, *Stochastic Approximation Algorithms and Applications*. New York, NY, USA: Springer-Verlag, 1997.

[23] J. Tsitsiklis, "Asynchronous stochastic approximation and Q-learning," *Machine Learning*, vol. 16, pp. 185–202, 1994.

[24] T. Jaakkola, M. Jordan, and S. Singh, "Convergence of stochastic iterative dynamic programming algorithms," in *Advances in Neural Information Processing Systems*, vol. 6, J. Cowan, G. Tesauro, and J. Alspector, Eds. San Francisco, CA, USA: Morgan Kaufmann Publishers, 1994, pp. 703– 710.

[25] M. G. Azar, R. Munos, M. Ghavamzadeh, and H. J. Kappen, "Speedy Q-learning," *Adv. Neural Inform. Processing Syst.*, vol. 24, pp. 2411–2419, 2011.

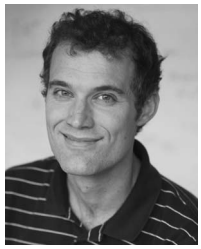
[26] C. Szepesvári, "The asymptotic convergence-rate of Q-learning," in *Advances in Neural Information Processing Systems*, vol. 10, M. Jordan, M. Kearns, and S. Solla, Eds. Cambridge, MA, USA: MIT Press, 1997, pp. 1064–1070.

[27] E. Even-Dar and Y. Mansour, "Learning rates for Q-learning," *J. Machine Learning Res.*, vol. 5, pp. 1–25, 2003.

[28] A. Gosavi, "On step sizes, stochastic shortest paths, survival probabilities in reinforcement learning," in *Proc. Winter Simul. Conf.*, S. J. Mason, R. R. Hill, L. Mönch, O. Rose, T. Jefferson, and J. W. Fowler, Eds., 2008, pp. 525–531.
[29] H. P. Simão and W. B. Powell, "Approximate dynamic programming for management of high-value spare parts," *J. Manufact. Technol. Manag.*, vol. 20, no. 2, pp. 147–160, 2009.
[30] A. Benveniste, M. Metivier, and P. Priouret, *Adaptive Algorithms and Stochastic Approximations*. New York, NY, USA: Springer-Verlag, 1990.
[31] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *J. Machine Learning Res.*, vol. 12, pp. 2121–2159, 2011.
[32] T. Schaul, S. Zhang, and Y. LeCun, "No more pesky learning rates," in *Proc. 30th Int. Conf. Machine Learning*, 2013, pp. 343–351.
[33] R. Sutton, "Adapting bias by gradient descent: An incremental version of delta-bar-delta," in *Proc. 10th Nat. Conf. Artif. Intell.*, 1992, pp. 171–176.
[34] A. R. Mahmood, R. S. Sutton, T. Degris, and P. M. Pilarski, "Tuning-free stepsize adaptation," in *Proc. IEEE Int. Conf. Acous., Speech, Signal Processing*, 2012, pp. 2121–2124.
[35] R. Stengel, *Optimal Control and Estimation*. New York, NY, USA: Dover Publications, 1994.
[36] D. P. Choi and B. Van Roy, "A generalized Kalman filter for fixed point approximation and efficient temporal-difference learning," *Discrete Event Dyn. Syst.*, vol. 16, pp. 207–239, 2006.
[37] A. George and W. B. Powell, "Adaptive stepsizes for recursive estimation with applications in approximate dynamic programming," *Machine Learning*, vol. 65, no. 1, pp. 167–198, 2006.
[38] M. Hutter and S. Legg, "Temporal difference updating without a learning rate," in *Advances in Neural Information Processing Systems*, vol. 20, J. C. Platt, D. Koller, Y. Singer, and S. Roweis, Eds. Cambridge, MA, USA: MIT Press, 2007, pp. 705–712.
[39] R. Sutton, C. Szepesvári, and H. Maei, "A convergent O(n) algorithm for off-policy temporal-difference learning with linear function approximation," *Adv. Neural Inform. Processing Syst.*, vol. 21, pp. 1609–1616, 2008.
[40] D. Silver, L. Newnham, D. Barker, S. Weller, and J. McFall, "Concurrent reinforcement learning from customer interactions," in *Proc. 30th Int. Conf. Machine Learning*, 2013, pp. 924–932.
[41] I. O. Ryzhov, P. I. Frazier, and W. B. Powell, "A new optimal stepsize for approximate dynamic programming," IEEE Trans. Autom. Control. [Online]. Available: http://arxiv.org/abs/1407.2676
[42] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. New York, NY, USA: Springer, 2001, ser. Statistics.
[43] M. Broadie, D. Cicek, and A. Zeevi, "General bounds and finite-time improvement for the Kiefer-Wolfowitz stochastic approximation algorithm," *Oper. Res.*, vol. 59, no. 5, pp. 1211–1224, 2011.
[44] P. Bickel and K. Doksum, *Mathematical Statistics - Basic Ideas and Selected Topics Volume 1*. Upper Saddle River, NJ, USA: Prentice Hall, 2001.
[45] A. George, W. B. Powell, and S. R. Kulkarni, "Value function approximation using multiple aggregation for multiattribute resource management," *J. Machine Learning Res.*, vol. 9, pp. 2079–2111, 2008.
[46] J. M. Nascimento and W. B. Powell, "Dynamic programming models and algorithms for the mutual fund cash balance problem," *Manag. Sci.*, vol. 56, no. 5, pp. 801–815, 2010.
[47] C. Cervellera, V. C. P. Chen, and A. Wen, "Optimization of a large-scale water reservoir network by stochastic dynamic programming with efficient state space discretization," *Eur. J. Oper. Res.*, vol. 171, no. 3, pp. 1139–1151, 2006.

Ilya O. Ryzhov (M'11) received the Ph.D. degree in operations research and financial engineering from Princeton University, Princeton, NJ, USA, in 2011.

He is an Assistant Professor in the Robert H. Smith School of Business, University of Maryland, College Park, MD, USA. He is a co-author of *Optimal Learning* (Wiley). His research deals with optimal learning and the broader area of stochastic optimization, with applications in non-profit fundraising and revenue management.



Peter I. Frazier received the Ph.D. degree in operations research and financial engineering from Princeton University, Princeton, NJ, USA, in 2009.

He is an Assistant Professor in the School of Operations Research and Information Engineering, Cornell University, Ithaca, NY, USA. His research interest is in dynamic programming and Bayesian statistics, focusing on the optimal acquisition of information and sequential design of experiments. He works on applications in simulation, optimization, medicine, and materials science.

Dr. Frazier received the AFOSR Young Investigator Award and an NSF CAREER Award.

Warren B. Powell (M'06) is a Professor in the Department of Operations Research and Financial Engineering, Princeton University, Princeton, NJ, USA, and Director of CASTLE Laboratory and the Princeton Laboratory for Energy Systems Analysis. He is the author of *Approximate Dynamic Programming: Solving the curses of dimensionality* and a co-author of *Optimal Learning* (Wiley). Currently, he is involved in applications in energy, transportation, health, and finance.