Proceedings of the 2011 Winter Simulation Conference
S. Jain, R. R. Creasey, J. Himmelspach, K. P. White, and M. Fu, eds.

STOCHASTIC POLICY SEARCH FOR VARIANCE-PENALIZED SEMI-MARKOV CONTROL

Abhijit Gosavi

219 Engineering Management
Missouri University of Science and Technology
Rolla, MO 65401, USA

Mandar Purohit

ESRI
380 New York Street
Redlands, CA 92373-8100

ABSTRACT

The variance-penalized metric in Markov decision processes (MDPs) seeks to maximize the average reward minus a scalar times the variance of rewards. In this paper, our goal is to study the same metric in the context of the semi-Markov decision process (SMDP). In the SMDP, unlike the MDP, the time spent in each transition is not identical and may in fact be a random variable. We first develop an expression for the variance of rewards in the SMDPs, and then formulate the VP-SMDP. Our interest here is in solving the problem without generating the underlying transition probabilities of the Markov chains. We propose the use of two stochastic search techniques, namely simultaneous perturbation and learning automata, to solve the problem; these techniques use stochastic policies and can be used within simulators, thereby avoiding the generation of the transition probabilities.

1 INTRODUCTION

Interest in variance-penalized control stems from the fact that a risk-averse controller may be more attracted to a solution that produces a low variance in the rewards with some sacrifice in the average reward (i.e., an average reward lower than its highest attainable value) than to one which produces the maximum expected reward but also produces a high variance in the rewards. A lower amount of variance makes the solution more stable in terms of its behavior, more predictable, and hence more reliable. This principle follows from the pioneering work of Markowitz (1952) in economics, where variance was first used to model variability or risk in returns obtained from financial instruments.

In a variance-penalized MDP (VP-MDP), the goal is to optimize the long-run average reward minus a positive scalar (θ) times the long-run variance of rewards. The VP-MDP has been studied in an excellent expository article by Filar, Kallenberg, and Lee (1989). The scalar is called the *risk-averseness* factor. Using a suitable value for this scalar, which is a fixed value depending on the level of risk-averseness of the controller, solutions (i.e., policies) can be obtained that are variance-averse. Clearly, the risk-averseness factor should be chosen carefully, since very small values for it can result in behavior imitating risk-neutrality, while very large values can result in solutions that produce low variance and low average rewards. Typically the controller desires low variability but not very low values for average rewards.

Filar, Kallenberg, and Lee (1989) show that under suitable assumptions, there exists a deterministic stationary policy for the VP-MDP that is optimal. Second, their work also demonstrates that the VP-MDP is more challenging than the risk-neutral MDP because it requires a quadratic program unlike the risk-neutral MDP which can be solved either via linear programming or dynamic programming (Bertsekas 1995). An important point to be noted here is that the quadratic programming approach, like the linear or dynamic programming approach, is model-based, i.e., it requires the transition probability (TP) model.

In this paper, we will focus on solving the Semi-MDP (SMDP). The difference between an MDP and an SMDP is that the time spent in each transition of the underlying Markov chains in an SMDP is a random variable; in the MDP, we assume that the time spent in each transition to be the same. The SMDP is thus more general than the MDP in terms of the problems it can handle. Problems in queueing and wireless communications are often SMDPs rather than MDPs.

Our aim here is to solve the variance-penalized SMDP (VP-SMDP) using a search over *stochastic* policies, i.e., policies in which actions are selected randomly in each state. We will use simulation and bypass the need to generate the TP model; in other words, our techniques will be *model-free*. The two solution techniques that we employ in this paper are: (i) simultaneous perturbation (Spall 1992) and (ii) learning automata (Wheeler and Narenda 1986). Next, we present an overview of how these techniques are useful in the context of this paper.

Simultaneous perturbation (SP) is a relatively new approach that works as an alternative to finite differences for the technique of steepest descent (or ascent) in non-linear programming. It is a form of stochastic gradient descent, which is a well-studied approach in the literature. The power of SP stems from its ability to solve an unconstrained optimization problem (under certain conditions) with M decision variables using only 2 function evaluations per iteration. This is in contrast to the finite differences (central differences) approach which requires $2M$ function evaluations per iteration. Hence for a problem with a large number of decision variables, the number of function evaluations, especially if made via simulation, can make finite differences computationally intractable, whereas SP remains tractable. Note that steepest descent can itself take several steps, and if each step requires $2M$ function evaluations, each of which takes a significant amount of time on the computer, the optimization process can become very slow. Thus, if the optimization process requires k iterations, SP needs only $2k$ function evaluations, whereas finite differences needs $2Mk$. In the SP approach for solving the VP-SMDP, we will treat the objective function as a function of the probability of selecting an action in a given state. These action-selection probabilities will serve as the decision variables and the variance-penalized metric will serve as the objective function.

The LA technique (Wheeler and Narenda 1986, Narendra and Thathachar 1989) is based on a reinforcement principle in which the algorithm begins with stochastic policies within a simulator. At the start, every action is selected with equal probability in every state. The value of the performance metric obtained from simulating a stochastic policy over a trajectory of state transitions is used as feedback to update the action-selection probabilities. At the end, the algorithm converges to a deterministic policy, which should be the optimal policy for the problem. The LA technique can be used within a simulator, and therefore does not require the TP model.

We now present a quick review of the relevant literature. The so-called policy gradient algorithm has been well-studied in the literature for solving MDPs (Baxter and Bartlett 2001) and SMDPs (Singh, Tadic, and Doucet 2007). This technique also uses the notion of gradient-descent with the action probability as a decision variable; however, in terms of how it computes the derivative, it is completely different from SP. Use of stochastic policies for solving MDPs without generating the TP model is also well-studied under the umbrella of actor-critics; see the classical texts on reinforcement learning for details (Bertsekas and Tsitsiklis 1996, Sutton and Barto 1998). A growing body of work that also uses stochastic policies but is primarily derivative-free is summarized in an excellent recent textbook (Chang et al. 2007). Finally, in the reinforcement learning community, risk-averse control in absence of the TP model has already attracted some interest (Gabor, Kalmar, and Szepesvari 1998, Geibel 2001, Heger 1994, Neueier and Mihatsch 1999, Gosavi 2007, Gosavi 2009, and Gosavi and Meyn 2010). But to the best of our knowledge, this is the first use of SP and LA for risk-penalized control.

The remainder of the this paper is structured as follows. Section 2 mathematically formulates the VP-SMDP. Sections 3 and 4 present the SP and the LA algorithms respectively. Section 5 presents some results from numerical experiments. Section 6 presents some concluding remarks.

2 PROBLEM FORMULATION

We begin with some notation in order to formulate our problem:

- $\mathscr{S}$: The set of states of the SMDP.

- $\mathscr{A}(i)$: The set of actions available in state i.
- $\hat{q}$: A stochastic (randomized) policy.
- $p_{\hat{q}}(i,a)$: Probability of selecting action $a \in \mathscr{A}(i)$, when the system is in state $i \in \mathscr{S}$ and policy $\hat{q}$ is pursued.
- $\mathbf{P_a}$: Transition probability matrix (TPM) when action a is selected in every state. Each element of this matrix is represented by $P_a(i,j)$.
- $\mathbf{P_{\hat{q}}}$: TPM associated with a policy $\hat{q}$. Each element of this matrix is represented by $P_{\hat{q}}(i,j)$.
- $\vec{\pi}_{\hat{q}}$: The vector of limiting (steady-state) probabilities associated with $\mathbf{P_{\hat{q}}}$; the ith element of this vector will be denoted by $\pi_{\hat{q}}(i)$.
- $r_{\hat{q}}$: The random reward in one transition when policy $\hat{q}$ is pursued.
- $t_{\hat{q}}$: The random time taken for transition when policy $\hat{q}$ is pursued.
- $r(i,a,j)$: The (non-random) immediate reward earned in one transition from i to j when action a is selected.
- $t(i,a,j)$: The time (possibly random) spent in one transition from i to j when action a is selected.
- $\bar{t}(i,a,j)$: The mean time spent in one transition from i to j when action a is selected.

It is assumed that the Markov chain associated with $\hat{q}$ is regular (Grinstead and Snell 1997). Then the limiting (or steady-state or invariant) probabilities of the Markov chain, denoted by $\pi_{\hat{q}}(i)$ for all $i \in \mathscr{S}$, exist. We can define the average of the immediate reward, the second moment of the immediate reward, and the average of the transition time as follows. For all $i \in \mathscr{S}, a \in \mathscr{A}(i)$,

$$\bar{r}(i,a) = \sum_{j \in \mathscr{S}} P_a(i,j) r(i,a,j); \bar{r}^2(i,a) = \sum_{j \in \mathscr{S}} P_a(i,j)(r(i,a,j))^2;$$

$$\bar{t}(i,a) = \sum_{j \in \mathscr{S}} P_a(i,j)\bar{t}(i,a,j), \text{ in which } \bar{t}(i,a,j) = \int_0^\infty y F_{i,a}(y) dy$$

where $F_{i,a}(.)$ denotes the cumulative distribution function of the transition time from state i when action a is chosen. Now, each element of the TPM associated with the policy $\hat{q}$ can be calculated as follows:

$$P_{\hat{q}}(i,j) = \sum_{a \in \mathscr{A}(i)} p_{\hat{q}}(i,a) \cdot P_a(i,j) \text{ for all } i \in \mathscr{S}, j \in \mathscr{S}. \tag{1}$$

Then the expected reward and the expected second moment of reward, associated with policy $\hat{q}$, are respectively are:

$$\mathsf{E}[r_{\hat{q}}] = \sum_{i \in \mathscr{S}} \pi_{\hat{q}}(i) \left(\sum_{a \in \mathscr{A}(i)} p_{\hat{q}}(i,a)\bar{r}(i,a) \right) \text{ and } \mathsf{E}[r^2_{\hat{q}}] = \sum_{i \in \mathscr{S}} \pi_{\hat{q}}(i) \left(\sum_{a \in \mathscr{A}(i)} p_{\hat{q}}(i,a)\bar{r}^2(i,a) \right).$$

The expected time associated with policy $\hat{q}$ is

$$\mathsf{E}[t_{\hat{q}}] = \sum_{i \in \mathscr{S}} \pi_{\hat{q}}(i) \left(\sum_{a \in \mathscr{A}(i)} p_{\hat{q}}(i,a)\bar{t}(i,a) \right).$$

The variance of the reward earned in one transition is

$$\mathsf{Var}(r_{\hat{q}}) = \mathsf{E}[r^2_{\hat{q}}] - (\mathsf{E}[r_{\hat{q}}])^2.$$

For the MDP, the objective function we study from Filar, Kallenberg, and Lee (1989) is:

$$\Phi_{\hat{q}} = \mathsf{E}[r_{\hat{q}}] - \theta \mathsf{Var}(r_{\hat{q}}), \tag{2}$$

where θ is the risk-averseness factor.

For the SMDP, we need to derive a suitable objective function. For the SMDP in the risk-neutral case, renewal theory is used to formulate the expressions for the long-run measures (Tijms 2003). For the variance in the renewal theorem, one possibility is to use the *cyclical* variance developed in Gosavi (2006), and the other is to use the *long-run* variance developed in Brown and Solomon (1975). We will examine both approaches next.

Following standard arguments in renewal theory, let C_n be the reward in the nth renewal cycle, Γ_n be the time consumed in the nth renewal cycle, the mean reward in any cycle be $\mathsf{E}[C_n] \equiv \mathsf{E}[C]$, and the mean time consumed in any cycle be $\mathsf{E}[\Gamma_n] \equiv \mathsf{E}[\Gamma]$. Using the approach in Gosavi (2006), we can use the cyclical variance in the renewal reward process to be $\lim_{t\to\infty} \frac{\sum_{n=1}^{N(t)}[C_n - \mathsf{E}[C]]^2}{t}$, where $N(t)$ denotes the number of renewal cycles by time t, which as shown in (Gosavi 2006) yields the following expression for the cycle variance in the renewal reward process:

$$\frac{\mathsf{E}[C^2] - (\mathsf{E}[C])^2}{\mathsf{E}[\Gamma]}.$$

As is standard in the risk-neutral case, we assume each transition of the embedded Markov chain of a given policy to be a renewal cycle, and then using this definition of variance we can obtain the following expression for the long-run variance of the reward in the SMDP for the policy:

$$\frac{\mathsf{E}[C^2] - (\mathsf{E}[C])^2}{\mathsf{E}[\Gamma]} = \frac{\mathsf{E}[r_{\hat{q}}^2] - (\mathsf{E}[r_{\hat{q}}])^2}{\mathsf{E}[t_{\hat{q}}]} \equiv \frac{\sigma^2(r_{\hat{q}})}{\mathsf{E}[t_{\hat{q}}]}. \tag{3}$$

Then using the above expression for the variance of the reward per unit time in an SMDP, we can construct our objective function similar to (2) as:

$$\Phi_{\hat{q}} = \frac{\mathsf{E}[r_{\hat{q}}]}{\mathsf{E}[t_{\hat{q}}]} - \theta \frac{\mathsf{E}[r_{\hat{q}}^2] - (\mathsf{E}[r_{\hat{q}}])^2}{\mathsf{E}[t_{\hat{q}}]}. \tag{4}$$

The other approach to measuring variance is to compute it with respect to the long-run expected reward in the renewal process, i.e., $\lim_{t\to\infty} \frac{\sum_{n=1}^{N(t)}[C_n - \rho\Gamma_n]^2}{t}$, where $\rho = \frac{\mathsf{E}[C]}{\mathsf{E}[\Gamma]}$. The above yields the following alternative expression for the variance in the reward of the renewal reward process (Brown and Solomon 1975) (see also Gosavi (2008) for an alternative proof):

$$\frac{\mathsf{E}[C^2]}{\mathsf{E}[\Gamma]} - 2\rho^2 \mathsf{E}[\Gamma] + \rho^2 \frac{\mathsf{E}[\Gamma^2]}{\mathsf{E}[\Gamma]}.$$

When we use this interpretation of the variance, we obtain the following definition of variance for the SMDP:

$$\Phi_{\hat{q}} = \frac{\mathsf{E}[r_{\hat{q}}]}{\mathsf{E}[t_{\hat{q}}]} - \theta \left[\frac{\mathsf{E}[r_{\hat{q}}^2]}{\mathsf{E}[t_{\hat{q}}]} - 2\frac{\mathsf{E}^2[r_{\hat{q}}]}{\mathsf{E}[t_{\hat{q}}]} + \frac{\mathsf{E}^2[r_{\hat{q}}]\mathsf{E}[t_{\hat{q}}^2]}{\mathsf{E}^3[t_{\hat{q}}]} \right]. \tag{5}$$

Filar, Kallenberg, and Lee (1989) show that associated with their definition of variance in the MDP, i.e., Eqn (2), one can construct the following quadratic program (QP) to solve the MDP:

$$\text{Maximize} \sum_{i\in\mathscr{S}} \sum_{a\in\mathscr{A}(i)} \bar{r}(i,a)x(i,a) - \theta \sum_{i\in\mathscr{S}} \sum_{a\in\mathscr{A}(i)} \bar{r}^2(i,a)x(i,a) + \theta \left[\sum_{i\in\mathscr{S}} \sum_{a\in\mathscr{A}(i)} \bar{r}(i,a)x(i,a) \right]^2 \text{ s.t.}$$

$$\text{for all } j \in \mathscr{S}, \sum_{a\in\mathscr{A}(j)} x(j,a) - \sum_{i\in\mathscr{S}} \sum_{a\in\mathscr{A}(i)} p(i,a,j)x(i,a) = 0; \tag{6}$$

$$\sum_{i\in\mathscr{S}}\sum_{a\in\mathscr{A}(i)} x(i,a) = 1; \tag{7}$$

$$\text{and } x(i,a) \geq 0 \quad \forall (i,a). \tag{8}$$

A significant result, Corollary 2.1 of Filar, Kallenberg, and Lee (1989), also proves that a stationary, optimal policy which is deterministic exists for the objective function defined in (2). This result exploits properties of the QP above. We can extend the result in Filar, Kallenberg, and Lee (1989) easily if we were to use the objective function in (4) for our SMDP. However, for the function defined in (5), it is not easy to extend the result essentially because it is not clear how to construct a suitable QP. In other words, (5) may not enjoy the desirable mathematical properties to make it tractable via a QP. Hence, we use (4). Our derivation of the QP is shown next.

Using renewal reward theory if $x(i,a)$ is the state-action frequency for state-action pair (i,a), the it can be shown from (4) that our objective would be to maximize:

$$\frac{\sum_{i\in\mathscr{S}}\sum_{a\in\mathscr{A}(i)}\bar{r}(i,a)x(i,a)}{\sum_{i\in\mathscr{S}}\sum_{a\in\mathscr{A}(i)}\bar{t}(i,a)x(i,a)} + \theta\frac{\sum_{i\in\mathscr{S}}\sum_{a\in\mathscr{A}(i)}\bar{r}^2(i,a)x(i,a)}{\sum_{i\in\mathscr{S}}\sum_{a\in\mathscr{A}(i)}\bar{t}(i,a)x(i,a)} - \theta\frac{\left[\sum_{i\in\mathscr{S}}\sum_{a\in\mathscr{A}(i)}\bar{r}(i,a)x(i,a)\right]^2}{\sum_{i\in\mathscr{S}}\sum_{a\in\mathscr{A}(i)}\bar{t}(i,a)x(i,a)}$$

subject to (6) and (8). If we introduce the following constraint:

$$\sum_{i\in\mathscr{S}}\sum_{a\in\mathscr{A}(i)} x(i,a)\bar{t}(i,a) = 1 \tag{9}$$

then, we can replace the above optimization problem by: Maximize

$$\sum_{i\in\mathscr{S}}\sum_{a\in\mathscr{A}(i)}\bar{r}(i,a)x(i,a) - \theta\sum_{i\in\mathscr{S}}\sum_{a\in\mathscr{A}(i)}\bar{r}^2(i,a)x(i,a) + \theta\left[\sum_{i\in\mathscr{S}}\sum_{a\in\mathscr{A}(i)}\bar{r}(i,a)x(i,a)\right]^2 \tag{10}$$

subject to (6), (9), and (8). (Note that a similar strategy is used for the risk-neutral SMDP to obtain a linear program (Tijms 2003).) The above is also a convex quadratic program (Filar, Kallenberg, and Lee 1989), and the constraint (9) produces a bounded polyhedron. Then we can argue as in Corollary 2.1 of Filar, Kallenberg, and Lee (1989) that there exists a stationary deterministic optimal policy for our variance-penalized SMDP (VP-SMDP). The reason for *not* using the objective function in (5) is that it is unclear if we can obtain such a result, since the resulting objective function will contain terms raised to powers higher than 2, thereby not allowing us to obtain a quadratic program.

3 SIMULTANEOUS PERTURBATION

As stated above, SP uses steepest descent (ascent in case of function maximization) for optimization. We will treat the variance-penalized metric in (4) as the objective function to be maximized. The decision variables will be the action-selection probabilities. It should be clear from the discussion in Section 1 that the objective function in (4) is clearly a function of the action-selection probabilities. For a given stochastic policy, which is defined by a given set of action probabilities, one can simulate the system for a long enough time such that a reliable estimate of the objective function in (4) can be obtained. This is the estimate that will be used as the value of a function at any given point in the decision space. In the algorithm below, $\lambda(i,a)$ will denote the probability of selecting action a in state i.

Steps in the SP Algorithm:

Set k, the iteration number, to 1, and let $k_{\max}$ denote the maximum number of iterations. In what follows, $m \equiv (i,a)$ where $i \in \mathscr{S}$ and $a \in \mathscr{A}(i)$. Let $\lambda(m)$ denote the mth decision variable for $m = 1,2,\ldots,M$ where M denotes the number of decision variables or the number of state-action pairs. Also, $\vec{\lambda} \equiv (\lambda(1),\lambda(2),\ldots,\lambda(M))$. Initialize $\lambda(m)$ to an arbitrary feasible point for all m. Let c^k denote the kth term of a decreasing sequence and μ^k the step size. More on selecting both of these below.

- **Step 1.** Using a Bernoulli distributed random variable $H^k(m)$ for $m = 1,2,\ldots,M$, whose two permissible *equally likely*, values are 1 and -1, compute $h^k(m)$ for all values of m, using
$$h^k(m) = H^k(m)c^k.$$
- **Step 2.** Calculate L^+ and L^- using the following formulas:
$$L^+ = L\left(\lambda^k(1)+h^k(1),\lambda^k(2)+h^k(2),\ldots,\lambda^k(M)+h^k(M|)\right),$$
and
$$L^- = L\left(\lambda^k(1)-h^k(1),\lambda^k(2)-h^k(2),\ldots,\lambda^k(M)-h^k(M)\right)$$
where $L(.)$ denotes the objective function in (4), estimated via simulation.
- **Step 3.** For $m = 1,2,\ldots,M$, obtain the value for the partial derivative using
$$\frac{\partial L(\vec{\lambda})}{\partial \lambda(m)} \approx \frac{L^+ - L^-}{2h^k(m)}.$$
- **Step 4.** For $m = 1,2,\ldots,M$, update $\lambda^k(m)$ using the following rule.
$$\lambda^{k+1}(m) \leftarrow \lambda^k(m) + \mu^k \frac{\partial L(\vec{\lambda})}{\partial \lambda(m)}.$$
- **Step 5.** Increment k by 1. If $k > k_{\max}$ stop; otherwise, return to Step 1.

Remark 1. Examples for the step-size parameters are: $c^k = \frac{0.1}{\sqrt{k}}$ and $\mu^k = 10/(100+k)$. Many other rules have been suggested in Spall (2003).

Also note that a takes values from $1,2,\ldots,|\mathscr{A}(i)|$. Hence, in the algorithm above, for each i, we will update only $|\mathscr{A}(i)|-1$ values of $\lambda(i,\cdot)$, i.e., only for $a = 1,2,\ldots,|\mathscr{A}(i)|-1$; the $|\mathscr{A}(i)|$th value will be updated as follows:
$$\lambda^{k+1}(i,|\mathscr{A}(i)|) = 1 - \sum_{a=1}^{|\mathscr{A}(i)|-1} \lambda^k(i,a).$$

In addition, we must ensure that the values of every component of $\vec{\lambda}$ are constrained in the interval $[0,1]$. This is achieved in Step 4 by projecting the updated value on the interval $[0,1]$:
$$\lambda^{k+1}(i,a) \leftarrow \Gamma_{[0,1]}\left[\lambda^k(i,a) + \mu^k \frac{\partial L(\vec{\lambda})}{\partial \lambda(i,a)}\right],$$
where $\Gamma_{[x,y]}[z] = x$ if $z \le x$, $\Gamma_{[x,y]}[z] = y$ if $z \ge y$, and $\Gamma_{[x,y]}[z] = z$ if $x < z < y$.

4 LEARNING AUTOMATA

The technique of LA uses the notion of simulating a long trajectory of states with stochastic policies whose action selection probabilities are gradually updated until they reach the optimal solution. At the start, the algorithm starts with the same probability of selecting any given action in every state. Actions are selected in each state using the action-selection probabilities. The objective function's value (ϕ) is estimated from the trajectory of states simulated and the rewards observed, the value is then transformed into a usable feedback value (β) that ranges from 0 to 1, and then the feedback is used to update the action-selection probabilities. Actions that result in positive feedback (β close to 1) have their action-selection probabilities increased, while actions that result in negative feedback (β close to 0) have their action-selection probabilities diminished.

The above is essentially the scheme described in Wheeler and Narenda (1986). Our goal here is to generate a mechanism to measure the value of the objective function of interest to us, i.e., the variance-penalized metric, from pertinent segments of a simulated trajectory and then use it as feedback.

Steps in the LA Algorithm for VP-SMDP:

Every time a state is visited in the simulator, the algorithm for the VP-SMDP needs to update the following quantities: (i) The time-averaged reward earned in the trajectory starting at the last visit to the current state and ending at the current state; this quantity is denoted by ρ. (ii) The average reward per transition of the Markov chain in the same trajectory; this quantity is denoted by ψ_1. (iii) The average of the squared reward per transition in the trajectory; this quantity is denoted by ψ_2.

Step 1 Set iteration count, k, to 0. Set η, the learning rate, to a small positive value less than 1. Initialize action probabilities $\lambda(i,a) = \frac{1}{|\mathscr{A}(i)|}$ for all $i \in \mathscr{S}$ and $a \in \mathscr{A}(i)$. Set the cumulative reward $C^r(i) = 0$, cumulative squared reward $C_2^r(i) = 0$ and cumulative time $C^t(i) = 0$ for all $i \in \mathscr{S}$. Also initialize to 0 the following quantities: Cumulative iterations, $C_k(i)$; cumulative reward earned in the system from start, $R^{current}$; cumulative squared reward earned in the system from start, $R_2^{current}$; simulation clock, $T^{current}$. Set suitable values to $\phi_{\max}$ and $\phi_{\min}$ (see Remarks below for how to select these values). Fix $a^*(i)$ for every $i \in \mathscr{S}$ to some distinguished action. Start the system simulation. Let the starting state be denoted by i.

Step 2 If this is the first visit to state i, go to Step 3. Otherwise, update the following quantities:

$$\rho \leftarrow \frac{R^{current} - C^r(i)}{T^{current} - C^t(i)}; \psi_1 \leftarrow \frac{R^{current} - C^r(i)}{k - C_k(i)}; \psi_2 \leftarrow \frac{R_2^{current} - C_2^r(i)}{k - C_k(i)}.$$

Then we obtain the raw feedback as follows:

$$\phi \leftarrow \rho - \theta \frac{\psi_2 - (\psi_1)^2}{\left(\frac{T^{current} - C^t(i)}{k - C_k(i)}\right)}, \tag{11}$$

and then convert that into the normalized feedback with:

$$\beta \leftarrow \frac{\phi - \phi_{\min}}{\phi_{\max} - \phi_{\min}}.$$

Step 2 Let $U(i)$ denote the action taken in the last visit to i. Update the action probabilities $\lambda(i,a^*(i))$ using:

$$\lambda(i,a^*(i)) \leftarrow \lambda(i,a^*(i)) + \Delta$$
$$\text{where } \Delta = \eta\beta I[U(i) = a^*(i)] - \eta\beta\lambda(i,a^*(i)),$$

where $I[.] = 1$ if the condition within brackets is satisfied and 0 otherwise. Then update all the other actions as follows: For all a other than $a^*(i)$,

$$\lambda(i,a) = \lambda(i,a) - \frac{\Delta}{|\mathscr{A}(i)| - 1},$$

so that sum of the action probabilities for state i is 1.

Step 3 Set $C^r(i) \leftarrow R^{current}, C_2^r(i) \leftarrow R_2^{current}, C^t(i) \leftarrow T^{current}$, and $C_k(i) \leftarrow k$

Step 4 With probability $\lambda(i,a)$, select an action a from $\mathscr{A}(i)$ and, set $U(i) \leftarrow a$. Simulate action a. Let the system state at the next decision epoch be j. Also let $t(i,a,j)$ denote the transition time and $r(i,a,j)$ denote the immediate reward earned in the transition resulting from selecting action a in state i.

Step 5 Set $R^{current} \leftarrow R^{current} + r(i,a,j), R_2^{current} \leftarrow R_2^{current} + r^2(i,a,j), T^{current} \leftarrow T^{current} + t(i,a,j)$.
Step 6 Set $i \leftarrow j$ and $k \leftarrow k+1$. If $k < k_{\max}$, return to step 2; otherwise stop.

Remark 2. The value of ϕ essentially denotes an estimate of the objective function from a finite trajectory. Hence, technically, $\phi_{\min}$ and $\phi_{\max}$ should respectively equal the minimum and maximum value that the objective function can assume. This requirement is typical of al LA algorithm based on the scheme in Wheeler and Narenda (1986). However, in practice what we found (see Purohit (2006) for more details) is that in the first few iterations, it was necessary to use a value much larger than the minimum value of ϕ for $\phi_{\min}$ and a value smaller than the maximum value of ϕ for $\phi_{\max}$. After a few iterations, we were able to gradually increase $\phi_{\max}$ and decrease $\phi_{\min}$ to their actual values; this was done in stages — somewhat like temperature reduction is simulated annealing. Without this, the value of β became very small making the optimization discouragingly slow.
Remark 3. Until the actual values of $\phi_{\max}$ and $\phi_{\min}$ are used, if ϕ turns out to be less than the value of $\phi_{\min}$ used or it turns out to be greater than the value of $\phi_{\max}$ used, we need to skip Steps 2 and 3 and jump to Step 4. If these steps are not skipped, we will end up with values of β either greater than 1 or less than 0 — both of these are unacceptable for the algorithm. Also, by skipping Steps 2 and 3, we ensure that the algorithm's updating is not tampered with, since the updating occurs in these steps. More importantly, this is no way affects the algorithm's asymptotic behavior, since it only ignores the updates of certain states for a finite time interval and $\phi_{\max}$ and $\phi_{\min}$ reach their correct values in the limit. This strategy of stage-wise change of the upper and lower limits of ϕ is performed to increase the speed of optimization.
Remark 4. We note that the following quantity in equation (11)

$$\frac{\psi_2 - (\psi_1)^2}{\left(\frac{T^{current} - C^t(i)}{k - C_k(i)}\right)}$$

essentially captures variance, as defined in (3). The denominator in the term above seeks to capture the expected time of each transition, and this scheme is borrowed from Gosavi, Das, and Sarkar (2004), which studied LA for risk-neutral SMDPs.

5 NUMERICAL EXPERIMENTS

In this section, we present results from some numerical experiments on some small problems with two states with two actions in each state. We chose to use the MDP model since it is simpler. We begin with a test case that we call $mdp1$. The transition probability and reward matrices for this test MDP be as follows. We will use $\mathbf{R}_a$ to denote the transition reward matrix when action a is selected in every state.

$$\mathbf{P}_1 = \begin{bmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{bmatrix}; \mathbf{P}_2 = \begin{bmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{bmatrix};$$

$$\mathbf{R}_1 = \begin{bmatrix} 6 & -5 \\ 7 & 12 \end{bmatrix}; \mathbf{R}_2 = \begin{bmatrix} 5 & 68 \\ -2 & 12 \end{bmatrix}.$$

The exhaustive enumeration method was used to evaluate the objective function corresponding to each deterministic policy. We will use $\zeta(i)$ to denote the action to be chosen in state i when a deterministic policy ζ is used for the system. Policy 3 is clearly the optimum policy for which $\mathsf{E}[r_\zeta] = 8.625$ and $\mathsf{Var}[r_\zeta] = 31.284375$. See Table 1 for details.

For SP, the objective function was evaluated by computing the steady-state probabilities of the underlying Markov chains, although simulation could also be used for the same purpose. SP converged to the optimal policy returning $\lambda(1,1) = 1$ and $\lambda(2,1) = 0$ after 50 iterations using $c^k = \frac{0.1}{\sqrt{k+1}}$ and $\mu = 0.01$. Note that in every iteration, one would require two function evaluations, which if performed via simulation can

Table 1: Scores for the different deterministic policies for *mdp*1 using $\theta = 0.2$.

Policy (ζ)	$\zeta(1)$	$\zeta(2)$	ϕ_ζ
Policy 1	1	1	-0.199837
Policy 2	2	1	-46.40768
Policy 3	1	2	2.368125
Policy 4	2	2	-26.559

consume a significant amount of time on the computer. For this algorithm to be useful, it must be used on problems whose transition probabilities are hard to find, where typically the only way to estimate the objective function would be via simulation. Generally, such a simulation would require numerous state transitions and then averaging over multiple replications. Hence, clearly an algorithm like SP must converge in a small number of iterations ($k_{\max}$) for it to be useful in practice.

For LA, we used $\eta = 0.05$, $\phi_{\max} = 10$, and $\phi_{\min} = -250$. The algorithm converged to the optimal policy, defined by $\lambda(1,1) = 1$ and $\lambda(2,1) = 0$ after about 2400 iterations. Note that although 2400 $>>$ 50, LA is in reality much faster than SP in terms of the time it takes on the computer. This is because each iteration in LA involves *only one* transition in the simulator, whereas the same in SP requires numerous transitions (as explained above).

It should be noted that for both algorithms the values of $\lambda(.)$ gradually make their way to 0 or 1. For instance, after the 1000 iterations, we had $\lambda(1,1) = 0.6908$ and $\lambda(2,1) = 0$ with LA. So clearly, all the values of λ may not converge simultaneously; furthermore, one could, in practice, set an upper and lower threshold for λ (e.g., 0.9 as the upper threshold and 0.1 as the lower threshold), and when λ falls below the lower threshold or above the upper threshold, its value can be assumed to be 0 or 1 respectively, and further updating can be stopped for that state-action pair. This is obviously not recommended unless the algorithm takes a very long time. The reason is: eventually the values of λ may in fact converge to a different policy than the one that would be obtained with setting thresholds. However, on large-scale problems, where time is of essence, such a practice is popular with LA algorithms, and may be required especially for SP which takes longer time.

We repeat the above exercise on another test case, which we call *mdp*2. The problem parameters are described as follows:

$$\mathbf{P}_1 = \begin{bmatrix} 0.2 & 0.8 \\ 0.7 & 0.3 \end{bmatrix}; \mathbf{P}_2 = \begin{bmatrix} 0.6 & 0.4 \\ 0.1 & 0.9 \end{bmatrix};$$

$$\mathbf{R}_1 = \begin{bmatrix} 6 & 9 \\ 11 & 14 \end{bmatrix}; \mathbf{R}_2 = \begin{bmatrix} 7 & 16 \\ 5 & 7 \end{bmatrix}.$$

Policy 1 is clearly the optimum policy for which $\mathsf{E}[r_\zeta] = 10.266667$ and $\mathsf{Var}[r_\zeta] = 4.728889$. See Table 2 for details.

Table 2: Scores for the different deterministic policies for *mdp*2 using $\theta = 0.5$.

Policy (ζ)	$\zeta(1)$	$\zeta(2)$	ϕ_ζ
Policy 1	1	1	7.9022
Policy 2	2	1	4.3481
Policy 3	1	2	6.6113
Policy 4	2	2	4.3168

For this case, SP converged in about 50 iterations using the same rules for c^k and μ, while LA converged after 3500 iterations using $\eta = 0.05$, $\phi_{\max} = 16$, and $\phi_{\min} = -20$. In both cases, we obtain the optimal policy with the ending values of $\lambda(1,1) = 1$ and $\lambda(2,1) = 1$. Computer programs were written in C, and tested on a LINUX machine using the gcc compiler. The run-time of each case was less than 1 second.

However, it needs to be pointed out that these are very small problems, and on larger problems, these methods, especially SP, should take a significantly longer period of time.

6 CONCLUSIONS

The VPMDP has been studied in the literature for many years now, beginning with the first formal treatment in Filar, Kallenberg, and Lee (1989). A simulation-based approach using Q-Learning for the same problem was studied in Gosavi and Meyn (2010). In this paper, our goal was to formulate the VP-SMDP and then present adaptations of two different algorithms based on stochastic search for solving the VP-SMDP. Both of these algorithms are simulation-based and hence do not require the determining of the system's transition probabilities, which, due to the curse of dimensionality, can become very difficult in complex systems. Much of this work comes from Purohit (2006), where the reader will also find numerical results which show that both algorithms converge to optimal solutions on other small problems and a problem from production systems.

Future work: Attempting to study analytical properties of a function that is evaluated via simulation in some sense is an "abstract ideal " (Spall 2003), since simulation-based optimization is typically used with those functions whose analytical properties are hard to ascertain. However, under the assumption that the simulator returns a near-perfect estimate, it may be possible to study some of the convergence properties of our algorithm using the remarkable result in He, Fu, and Marcus (2003), which bypasses the need for showing differentiability — a key hurdle in this context. Convergence properties of the LA algorithm will require showing that the VP-SMDP has a unique Nash equilibrium in the game-theoretic setting established in Wheeler and Narenda (1986) for the analysis of ergodic control problems. Empirical work in Purohit (2006) shows that it may be possible to construct a simple two-state VP-MDP in which the Nash equilibria are not unique and yet in practice the algorithm converges to the optimal solution. Clearly, thus, the convergence analysis of this algorithm remains an open issue that requires further study.

ACKNOWLEDGMENTS

We thank Aykut F. Kahraman (M & T Bank, Buffalo, NY) and Wayne F. Bialas (Professor Emeritus at the University at Buffalo, SUNY) for helpful discussions on the convergence arguments of SP and Nash equilibria in games.

REFERENCES

Baxter, J., and P. Bartlett. 2001. "Infinite-horizon policy-gradient estimation". *Journal of Artificial Intelligence* 15:319–350.

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

Bertsekas, D. P. 1995. *Dynamic Programming and Optimal Control*. Belmont, Massachusetts: Athena Scientific.

Brown, M., and H. Solomon. 1975. "A second-order approximation for the variance of a renewal reward process". *Stochastic Processes and their Applications* 3:301–314.

Chang, H., M. Fu, J. Hu, and S. Marcus. 2007. *Simulation-based algorithms for Markov decision processes*. NY: Springer.

Filar, J., L. Kallenberg, and H. Lee. 1989. "Variance-penalized Markov decision processes". *Mathematics of Operations Research* 14(1):147–161.

Gabor, Z., Z. Kalmar, and C. Szepesvari. 1998. "Multi-criteria reinforcement learning". In *Proc. of 15th International Conf. on Machine Learning*, 197–205: Morgan Kaufmann.

Geibel, P. 2001. "Reinforcement Learning via bounded risk". In *ICML01*, 162–169: Morgan Kaufman.

Gosavi, A. 2006. "A risk-sensitive approach to total productive maintenance". *Automatica* 42:1321–1330.

Gosavi, A. 2007. "Adaptive Critics for Airline Revenue Management". In *Proceedings of 18th Annual Conference of the Production and Operations Management Society*.

Gosavi, A. 2008, Dec. “On Step Sizes, Stochastic shortest paths, and survival probabilities in Reinforcement Learning”. In *Proceedings of the 2008 Winter Simulation Conference*, edited by S. J. Mason, R. R. Hill, L. Moench, O. Rose, T. Jefferson, and J. W. Fowler, 525–531. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Gosavi, A. 2009, December. “Reinforcement Learning for model building and variance-penalized control”. In *Proceedings of the 2009 Winter Simulation Conference*, edited by M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin, and R. G. Ingalls. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Gosavi, A., T. Das, and S. Sarkar. 2004. “A Simulation-Based Learning Automata Framework for Solving Semi-Markov Decision Problems Under Long-run Average Cost”. *IIE Transactions* 36:1–11.

Gosavi, A., and S. Meyn. 2010. “Value Iteration on Two Time-Scales for Variance-Penalized Markov Control”. In *Proceedings of the 2010 Industrial Engineering Research Conference A. Johnson and J. Miller, eds.* IIE Press.

Grinstead, C. M., and J. L. Snell. 1997. *Introduction to Probability*. 2nd ed. http://www.ams.org: American Mathematical Society.

He, Y., M. Fu, and S. Marcus. 2003. “Convergence of Simultaneous perturbation stochastic approximation for non-differentiable optimization”. *IEEE Transactions for Automatic Control* 48(8):1459–1463.

Heger, M. 1994. “Considerations of risk in reinforcement learning”. In *Proceedings of 11th International Conference on Machine Learning*, 105–111: Morgan Kaufmann.

Markowitz, H. 1952. “Portfolio Selection”. *Journal of Finance* 7(1):77–91.

Narendra, K., and M. Thathachar. 1989. *Learning Automata: An Introduction*. Prentice-Hall.

Neueier, R., and O. Mihatsch. 1999. “Risk-sensitive Reinforcement Learning”. In *Advances in NIPS (vol 2)*, 1031–1037: MIT Press.

Purohit, M. 2006. “Stochastic approximation methods for risk-sensitive control of discrete-event systems”. Unpublished Master’s thesis, SUNY, Buffalo.

Singh, S., V. Tadic, and A. Doucet. 2007. “A policy-gradient method for semi-Markov decision processes with application to call admission control”. *European Journal of Operational Research* 178:808–818.

Spall, J. 1992. “Multivariate stochastic approximation using a simultaneous perturbation gradient approximation”. *IEEE Transactions on Automatic Control* 37(3):332–341.

Spall, J. 2003. *An Introduction to Stochastic Search and Optimization*. Wiley.

Sutton, R., and A. G. Barto. 1998. *Reinforcement Learning*. Cambridge, Massachusetts: The MIT Press.

Tijms, H. 2003. *A first course in stochastic models*. England: Wiley.

Wheeler, R. M., and K. S. Narenda. 1986. “Decentralized learning in finite Markov chains”. *IEEE Transactions on Automatic Control* 31 (6): 519–526.

AUTHOR BIOGRAPHIES

ABHIJIT GOSAVI joined the Missouri University of Science and Technology in 2008 as an Assistant Professor. His research interests broadly include simulation-based optimization, Markov decision processes, preventive maintenance, and quality control. He has authored the textbook *Simulation-based Optimization*, published by Springer in 2003. His recent publications have appeared in journals such as the *INFORMS Journal on Computing*, *Automatica*, and the *European Journal of Operational Research*. He has a Ph.D. in industrial engineering from the University of South Florida, an M. Tech from the Indian Institute of Technology, Madras in Mechanical Engineering, and a B.E. in Mechanical Engineering from Jadavpur University. His email address is gosavia@mst.edu.

MANDAR PUROHIT is a Data Support Analyst at ESRI which develops geographic information system (GIS) solutions. He has an M.S. in Operations Research from the University at Buffalo (SUNY) and a B.E. in Mechanical Engineering from the University of Pune. His research interests include artificial intelligence and operations research. His email address is mv.purohit@gmail.com.