International Journal of General Systems
Vol. 00, No. 00, Month 200x, 1–25

RESEARCH ARTICLE

Variance-Penalized Markov Decision Processes: Dynamic Programming and Reinforcement Learning Techniques

Abhijit A. Gosavi
219 Engineering Management Building
Missouri University of Science and Technology
Rolla, MO 65401.
Tel: (573)341-4624
gosavia@mst.edu

(*Received 00 Month 200x; final version received 00 Month 200x*)

ISSN: 0308-1079 print/ISSN 1563-5104 online
© 200x Taylor & Francis
DOI: 10.1080/0308107YYxxxxxxxx
http://www.informaworld.com

In control systems theory, the Markov decision process (MDP) is a widely used optimization model involving selection of the optimal action in each state visited by a discrete-event system driven by Markov chains. The classical MDP model is suitable for an agent/decision-maker interested in maximizing expected revenues, but does not account for minimizing variability in the revenues. An MDP model in which the agent can maximize the revenues while simultaneously controlling the variance in the revenues is proposed. This work is rooted in machine learning/neural network concepts, where updating is based on system feedback and step sizes. First a Bellman equation for the problem is proposed. Thereafter, convergent dynamic programming and reinforcement learning techniques for solving the MDP are provided along with encouraging numerical results on a small MDP and a preventive maintenance problem.

Keywords: Variance-penalized MDPs; dynamic programming; risk penalties; reinforcement learning; Bellman equation,

1. Introduction

The theory of Markov decision processes (MDPs) has attracted much attention within the operations research (Bertsekas and Tsitsiklis, 1996) and artificial intelligence (Sutton and Barto, 1998) communities for controlling discrete-event systems. The MDP and its variants can be used for systems that have Markov chains driving their state transitions, and as such it is a widely studied topic within the domain of systems and control. In dynamic systems, the state evolves over time, and decisions made in each time step affect the trajectory adopted by the system. Where decision-making has to be performed over each time step and there is randomness in how the state changes, the MDP is a widely used model. The Bellman equation is the most frequently used formalism for solving such problems.

MDPs are thus problems of sequential decision making in which the optimizer is expected to determine the optimal action in each time step in order to maximize or minimize a given objective function. In such problems, the decision variable is the action chosen in each state, and the objective function is typically a function of the net rewards earned (or costs incurred) over all the time steps encountered in the time horizon of interest. In systems driven by Markov chains, the probability of transition to the next state depends only on the current state and the action chosen in the current state. Despite this limitation, Markovian problems abound in the industry with applications from bio-informatics and voice recognition to economics and production/service management. Problems where the transitions are dependent on two or more steps can sometimes be converted into single-step problems by augmenting the state space (Ross, 1997).

The classical MDP is studied to maximize the average or discounted reward earned from the state transitions of the underlying system. The average reward criterion under the infinite time horizon seeks to maximize the expected value of the reward earned *per transition*. In this paper, the focus is on restraining the variability of the reward in every transition from its expected value, i.e., the so-called "risk" in the rewards, within an infinite time horizon setting.

A significant body of literature already exists on this topic. "Risk-sensitive" control, which employs an exponential utility function, and other forms of control that account for "downside-risk," "value-at-risk," and "HARA utility" have been studied in the literature (Howard and Matheson, 1972; Cavazos-Cadena and Fernandez-Gaucherand, 1999; Bouakiz and Kebir, 1995; Guo, 2007; White, 1993; Wu and Lin, 1999; Filar et al., 1995; Boda and Filar, 2005; Lim and Zhou, 2001). Also, there exists work on using variance as a measure of risk (Sobel, 1982; Filar et al., 1989; Gosavi, 2006, 2011). In the artificial intelligence (machine learning) community, there has been related work that seeks to refine the computational aspects of the solution techniques (Heger, 1994; Littman and Szepesvari, 1996; Sato and Kobayashi, 2001; Mihatsch and Neuneier, 2002; Geibel and Wysotzki, 2005).

Some recent work on incorporating risk within dynamic programming and reinforcement learning includes Huang et al. (2013), Niv et al. (2012), Gosavi et al. (2011), and Sakaguchi and Ohtsubo (2010).

The goal in this paper is to maximize the difference between the average of the reward per transition and a *penalty factor* (a positive scalar, θ) times the classical long-run variance of the reward per transition (i.e., per step of the Markov chain). The penalty factor θ serves to quantify the degree of risk aversion, and the resulting metric (average reward minus θ times the variance) serves to identify policies that seek a compromise: a high (not necessarily the highest) average reward, but a low variance. This variance-penalized (or variance-adjusted) metric has been studied in Filar et al. (1989) and Sato and Kobayashi (2001). The central aim in this work is to develop a Bellman optimality equation for the problem and solve it via dynamic programming and also reinforcement learning — in particular, via some form of Q-Learning (Watkins, 1989).

Filar et al. (1989) set up the problem as a convex quadratic program. Although their approach does not yield a Bellman equation, it provides interesting insights on the nature of the problem, one of which is that *the problem has a deterministic stationary optimal solution* (policy). The approach of Sato and Kobayashi (2001) presents a version of the Bellman equation, but seeks to use a policy gradient algorithm for solution purposes.

A Bellman optimality equation is developed for the problem at hand, and thereafter it is shown that it can produce the desired solution under some conditions. This is followed by the development of a value iteration technique that solves the problem. Finally, a reinforcement learning (RL) algorithm, based on the value iteration technique, which can be implemented within simulators of the system in cases where the transition probabilities of the underlying Markov chains are not available, is presented.

The key contribution here is in showing for the first time that solutions to the Bellman equation for the variance-penalized problem have desirable qualities, as well as in deriving a dynamic programming and an RL technique for solution purposes. The paper concludes with a numerical exercise, via a preventive maintenance problem, where the proposed algorithms generate optimal or near-optimal solutions.

The rest of this article is organized as follows. Section 2 presents the Bellman equation for the problem, along with an analysis of the equations. Section 3 presents a value iteration algorithm along with its convergence proof. The RL algorithm is discussed in Section 4. Numerical tests with are provided in Section 5. Section 6 provides some concluding remarks.

2. Variance-Penalized MDPs

While the MDP is a widely used model, variants of the MDP that track risk along with expected revenues are less commonly studied in the literature. Hence, the need for studying such a model is first motivated. In decision making when revenues are involved it is quite common for the decision making to have a psychological threshold for the revenues. When revenues fall below this threshold, the decision-maker perceives that as an undesirable/risky occurrence. Therefore, when faced with the choice between high expected revenues with large amounts of variability, which includes variability of returns both above and below psychological thresholds, and somewhat lower expected revenues with lower variability, a risk-averse manager prefers the latter choice. This is especially well-known in the world of investment finance; it has also been studied in revenue management (Mitra and Wang, 2005;

Barz and Waldmann, 2007). It appears thus that a risk-averse decision-maker who operates on MDP models would find value in Bellman equations that account for risk in addition to maximizing expected revenues.

To this end, a version of a Bellman equation that penalizes variance is developed. Note that the classical version of the Bellman equation ignores variance. The field of artificial intelligence has extensively studied step-size-based (or learning-rate-based) algorithms, also called learning-based algorithms, that seek to update the value function (or usually the action-value function) slowly using step-sizes or learning rates. Building on this notion, a Bellman equation, which is based on action-value function (also called Q-function in artificial intelligence), rather than the value function, is constructed, and then a value iteration technique that updates the action-value function slowly via step-sizes to its optimal value is developed. The value iteration algorithm does not have any noise, typically found in artificial intelligence, and hence the step sizes can be constant and in fact quite large (very close to 1). Thereafter, a simulation-based counterpart of this algorithm that can work in a noisy simulation environment is developed. In the simulation environment, the transition probabilities will not be needed; however, the step-sizes will have to follow the standard conditions of stochastic approximation (i.e., in particular, they must be of the standard decreasing nature).

Figure 1 shows the main theme underlying the simulation-based algorithm proposed. When in the current state (i), the agent selects an action (a), transitions to the next state (j) as a result, gathers feedback about the immediate reward and the risk (variance), and updates in its database the action values (Q-factors) of the state-action pair (i, a) that it just left behind in its trajectory. After repeated trials and errors, it "learns" the optimal policy, i.e., the optimal action for every state in the system.

2.1 *Notation and Background*

Some notation and definitions that will be need are first provided. Analysis will be limited to problems in which the state space and action space are finite. Also, only stationary policies will be considered in the analysis. Let $\mathcal{S}$ denote the finite set of states, $\mathcal{A}(i)$ the finite set of actions permitted in state i, and $\mu(i)$ the action chosen in state i when policy μ is pursued, where $\cup_{i\in\mathcal{S}}\mathcal{A}(i) = \mathcal{A}$. The set of permissible stationary policies for the problem is denoted $\mathcal{P}$. Further let $r(.,.,.) : \mathcal{S}\times\mathcal{A}\times\mathcal{S} \to \mathbb{R}$ denote the one-step immediate reward and $p(.,.,.) : \mathcal{S}\times\mathcal{A}\times\mathcal{S} \to [0,1]$

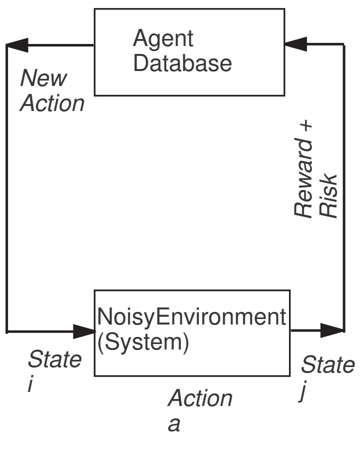

Figure 1. Schematic of an intelligent agent that gathers feedback on the risk and reward and learns the optimal policy in a noisy environment after repeated trial and errors.

denote the associated transition probability. Then the *expected* immediate reward earned in state i when action a is chosen in it can be expressed as: $\bar{r}(i,a) = \sum_{j=1}^{|\mathcal{S}|} p(i,a,j)r(i,a,j)$. E_μ will denote the expectation operator induced by the policy μ.

Definition 2.1: The long-run average (expected) reward of a policy μ starting at state i is:

$$\rho_\mu(i) \equiv \lim_{k\to\infty} \frac{1}{k}\mathsf{E}_\mu\left[\sum_{s=1}^{k} r(x_s,\mu(x_s),x_{s+1})|x_1 = i\right].$$

If the Markov chain of the policy is regular (Grinstead and Snell, 1997), the long-run average reward does not depend on the starting state, i.e., $\rho_\mu(i) = \rho_\mu$ for all $i \in \mathcal{S}$. Throughout this paper it will be assumed that every policy induces a Markov chain that is regular. The expected immediate variance in state i when policy μ is pursued will be defined as follows. For any $i \in \mathcal{S}$,

$$\bar{v}_\mu(i) = \sum_{j\in\mathcal{S}} p(i,\mu(i),j)[r(i,\mu(i),j) - \rho_\mu]^2, \tag{1}$$

where x_s denotes the state in the sth time step/transition. Te long-run variance of the reward of a policy following Filar et al. (1989).

Definition 2.2: The long-run variance of the reward of a policy μ starting at state i is given by:

$$\sigma^2_\mu(i) \equiv \lim_{k\to\infty} \frac{1}{k}\mathsf{E}_\mu\left[\sum_{s=1}^{k}(r(x_s,\mu(x_s),x_{s+1}) - \rho_\mu)^2 \middle| x_1 = i\right].$$

Like the average reward, variance can be shown to be independent of the starting state when the Markov chain of the policy is regular. In a classical MDP, the goal is to identify the policy that maximizes the total discounted reward over an infinite time horizon or the average reward over an infinite time horizon Bertsekas (1995). In a variance-penalized MDP, one seeks to maximize over all $\mu \in \mathcal{P}$, the variance-penalized score, i.e.,

$$\max_{\mu\in\mathcal{P}} \phi_\mu \equiv \rho_\mu - \theta\sigma^2_\mu, \tag{2}$$

where θ is a positive scalar that usually takes small values, e.g., 0.1.

It was shown in Filar et al. (1989) that a quadratic program (QP) can be used to solve the problem described above. The QP is as follows:

$$\text{Maximize} \sum_{i\in\mathcal{S}}\sum_{a\in\mathcal{A}(i)} \left[\bar{r}(i,a) - \theta\bar{r}^2(i,a)\right] y(i,a) + \theta\left[\sum_{i\in\mathcal{S}}\sum_{a\in\mathcal{A}(i)} \bar{r}(i,a)y(i,a)\right]^2$$

$$\text{subject to} \sum_{a\in\mathcal{A}(j)} y(j,a) - \sum_{i\in\mathcal{S}}\sum_{a\in\mathcal{A}(i)} p(i,a,j)y(i,a) = 0 \text{ for all } j \in \mathcal{S},$$

$$\sum_{i\in\mathcal{S}}\sum_{a\in\mathcal{A}(i)} y(i,a) = 1, \text{ and } y(i,a) \geq 0 \text{ for all } i \in \mathcal{S} \text{ and } a \in \mathcal{A}(i).$$

A key result, Corollary 2.1 in Filar et al. (1989), establishes that when Markov chains of all policies are regular, there exists a deterministic optimal (stationary) policy.

2.2 *Bellman Equations*

As stated in the introduction, the interest in this paper is in dynamic programming solutions. To this end, an appropriate Bellman optimality equation and the Bellman equation for a given policy for the problem at hand are proposed. This idea has been studied previously but has not been analyzed in depth before. In particular, Sato and Kobayashi (2001) develop a form of the variance-penalized Bellman equation but use it in the so-called policy gradient context, and do not present any mathematical analysis of the equation. Mihatsch and Neuneier (2002) show the relationship between variance penalties and the exponential utility functions (used in MDPs as discussed above), which are popular in economic risk because of the pleasant mathematical properties they enjoy; these properties can be proved via the famous Arrow-Pratt metric of risk aversion. They also explain the difficulties in using exponential utility functions in MDPs and develop a new metric for risk-sensitivity that is unrelated to variance. Gosavi (2007, 2009) and Gosavi and Meyn (2010) present some computational algorithms and numerical results, but do not provide any analysis of the Bellman equation.

Two key transformations, whose domain and range consists of functions $h\colon \mathcal{S} \to \mathbb{R}$, are now defined:

Definition 2.3: Given a function $h : \mathcal{S} \to \mathbb{R}$, we define the transformation L_μ as:

$$L_\mu(h(i)) = \sum_{j\in\mathcal{S}} p(i,\mu(i),j)\left[r(i,\mu(i),j) - \theta(r(i,\mu(i),j)-\rho)^2 + h(j)\right] \qquad \forall i \in \mathcal{S},$$

and the transformation G_ρ as:

$$G_\rho(h(i)) = \max_{a\in\mathcal{A}(i)}\left[\sum_{j\in\mathcal{S}} p(i,a,j)\left[r(i,a,j) - \theta(r(i,a,j)-\rho)^2 + h(j)\right]\right] \qquad \forall i \in \mathcal{S}.$$

Note that throughout this paper, L_μ, which depends on the value of ρ, will be understood to employ $\rho = \rho_\mu$; but in G_ρ, ρ will be any scalar. The following notation will also be employed.

$$L_\mu^{m+1}(h(i)) \equiv L_\mu(L_\mu^m(h(i))) \quad \forall i \in \mathcal{S}, m = 1,2,\ldots,;$$

$$L_\mu^1(.) \equiv L_\mu(.); \qquad L_\mu^0(h(i)) \equiv h(i) \quad \forall i \in \mathcal{S}.$$

Two elementary lemmas, whose proofs can be derived easily from their counterparts for the classical risk-neutral case (see e.g., Bertsekas (1995)) by replacing $\bar{r}(x_s,\mu(x_s))$ with $\bar{r}(x_s,\mu(x_s)) - \theta\bar{v}_\mu(x_s)$, are presented now.

Lemma 2.4: *Given a function* $h : \mathcal{S} \to \mathbb{R}$

$$L_\mu^k h(i) = \mathsf{E}_\mu\left[h(x_{k+1}) + \sum_{s=1}^{k}\left[\bar{r}(x_s,\mu(x_s)) - \theta\bar{v}_\mu(x_s)\right]\Bigg| x_1 = i\right],$$

for all values of $i \in \mathcal{S}$,

Lemma 2.5: *The transformation L_μ is monotonic, i.e., given two vectors, J and J', which satisfy the relation $J(i) \leq J'(i)$ for every $i \in \mathcal{S}$ the following is true for any positive integral k: $L^k_\mu J(i) \leq L^k_\mu J'(i)$ for every $i \in \mathcal{S}$.*

The following is a Bellman equation for a given policy.

Proposition 2.6: *Consider a policy μ. If a scalar $\phi \in \mathbb{R}$ and a function $h : \mathcal{S} \to \mathbb{R}$ satisfy for all $i \in \mathcal{S}$:*

$$\phi + h(i) = \sum_{j \in \mathcal{S}} p(i, \mu(i), j) \left[r(i, \mu(i), j) - \theta(r(i, \mu(i), j) - \rho_\mu)^2 + h(j) \right], \tag{3}$$

then ϕ will equal the variance-penalized score associated with the policy μ.

The above, i.e., Equation (3), can be viewed as the Bellman equation for a given policy. The proof can be worked out in a manner analogous to its risk-free counterpart, and is hence relegated to Appendix A.

Let $\mathcal{X}$ denote the finite set of distinct (scalar) values of ρ that equal the average reward of all the different permissible (deterministic) policies.

Assumption 2.7 Assume that a scalar ϕ_* and a function $J : \mathcal{S} \to \mathbb{R}$ exist that solve the following equation:

$$\phi_* + J(i) = \max_{\rho \in \mathcal{X}} \left[\max_{a \in \mathcal{A}(i)} \left[\sum_{j \in \mathcal{S}} p(i, a, j) \left[r(i, a, j) - \theta(r(i, a, j) - \rho)^2 + J(j) \right] \right] \right], \tag{4}$$

for every $i \in \mathcal{S}$ in a way such that there exists a maximizer ρ_*, solving (4) for every $i \in \mathcal{S}$. Further, ρ_* equals the average reward of the policy μ_* defined by

$$\mu_*(i) \in \arg\max_{a \in \mathcal{A}(i)} \left[\sum_{j \in \mathcal{S}} p(i, a, j) \left[r(i, a, j) - \theta(r(i, a, j) - \rho_*)^2 + J(j) \right] \right] \tag{5}$$

for every $i \in \mathcal{S}$.

Proposition 2.8: *Under Assumption 2.7, μ_* is an optimal policy for the problem, i.e., μ_* solves the problem defined in (2).*

The above result relies on using a solution of Equation (4) that will pick a value of ρ that not only solves the equation for *every* value of $i \in \mathcal{S}$ but in addition equals the average reward of the policy that satisfies (5). That a solution of this nature exists for Equation (4) is *assumed*; no attempt is made to show the existence of a solution. However, it will be shown that should such a solution exist, it is the optimal solution. After proving this, the next goal is to develop an algorithm that generates a function that not only solves Equation (4) but also ensures that the policy whose average reward is used in Equation (4) for solution satisfies (5). The algorithm is described in the next section. The proof of the above result follows.

Proof: From the assumptions, it follows that the function J and scalars, ϕ_* and ρ_*, solve the following equation for every $i \in \mathcal{S}$:

$$\phi_* + J(i) = \sum_{j \in \mathcal{S}} p(i, \mu_*(i), j) \left[r(i, \mu_*(i), j) - \theta(r(i, \mu_*(i), j) - \rho_{\mu_*})^2 + J(j) \right]. \tag{6}$$

Then Proposition 2.6 implies that ϕ_* is the variance-penalized score of the policy μ_*. It will be shown that any given policy, μ, will produce a variance-penalized score lower than or equal to ϕ_*. Let e denote a column vector of ones. It will be first proved that:

$$k\phi_* e + J \geq L_\mu^k(J) \text{ for any integer } k. \tag{7}$$

Now from Equation (4) and from the definition of G_ρ, using $\rho = \rho_\mu$ in Definition 2.3, for every $i \in \mathcal{S}$,

$$\phi_* + J(i) \geq \left[\max_{a \in \mathcal{A}(i)} \left[\sum_{j \in \mathcal{S}} p(i,a,j) \left[r(i,a,j) - \theta(r(i,a,j) - \rho_\mu)^2 + J(j) \right] \right] \right]$$

$$= G_{\rho_\mu}(J(i)).$$

From Definition 2.3, it follows that for any vector J,

$$G_{\rho_\mu}(J) \geq L_\mu(J), \text{ which from the above implies that } \phi_* e + J \geq L_\mu(J),$$

thus proving (7) holds for $k = 1$. Assuming it is true for $k = m$: $L_\mu^m(J) \leq m\phi_* e + J$. Using the fact that L_μ is monotonic from Lemma 2.5, the induction is completed as follows:

$$\begin{aligned} L_\mu(L_\mu^m)(J) &\leq L_\mu(m\phi_* e + J) \\ &= m\phi_* e + L_\mu(J) \leq (m+1)\phi_* e + J. \end{aligned}$$

Using Lemma 2.4 and inequality (7), one has that for any $i \in \mathcal{S}$:

$$\mathsf{E}_\mu \left[J(x_{k+1}) + \sum_{s=1}^{k} \left[\bar{r}(x_s, \mu(x_s)) - \theta \bar{v}_\mu(x_s) \right] \middle| x_1 = i \right] \leq k\phi_* + J(i).$$

Dividing both sides by k and taking the limit with $k \to \infty$, one has that:

$$\phi_\mu = \lim_{k \to \infty} \frac{1}{k} \mathsf{E}_\mu \left[\sum_{s=1}^{k} \left[\bar{r}(x_s, \mu(x_s)) - \theta \bar{v}_\mu(x_s) \right] \middle| x_1 = i \right] \leq \phi_*.$$

□

2.3 *A Q-factor version of the Bellman optimality equation*

For computational purposes, a version of Equation (4) that uses a Q-factor version of the Bellman equation proposed above (Sutton and Barto, 1998) will be used. The Q-factor is defined as follows: For all (i, a) pairs,

$$Q(i,a) = \sum_{j \in \mathcal{S}} p(i,a,j) \left[r(i,a,j) - \theta(r(i,a,j) - \rho_*)^2 - \phi_* + J(j) \right], \tag{8}$$

where J, ϕ_*, and ρ_* are defined as in Assumption 2.7. Using the above definition, the Bellman equation, i.e., Equation (4) can be written as:

$$J(i) = \max_{a \in \mathcal{A}(i)} Q(i,a), \tag{9}$$

which when combined with the definition in Equation (8) leads one to Q-factor version of the Bellman equation:

$$Q(i,a) = \sum_{j \in \mathcal{S}} p(i,a,j) \left[r(i,a,j) - \theta(r(i,a,j) - \rho_*)^2 - \phi_* + \max_{b \in \mathcal{A}(j)} Q(j,b) \right]. \tag{10}$$

Assumption 2.7 can now be restated in terms of Q-factors as:

Assumption 2.9 Assume that scalars ϕ_* and ρ_* and a function $Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ exist that solve the following equation:

$$Q(i,a) = \sum_{j \in \mathcal{S}} p(i,a,j) \left[r(i,a,j) - \theta(r(i,a,j) - \rho_*)^2 - \phi_* + \max_{b \in \mathcal{A}(j)} Q(j,b) \right],$$

such that ρ_* equals the average reward of the policy defined as follows:

$$\mu_*(i) \in \arg\max_{a \in \mathcal{A}(i)} Q(i,a).$$

Proposition 2.8 can now be stated in terms of Q-factors as follows:

Proposition 2.10: *Under Assumption 2.9, μ_* is an optimal policy for the problem defined in (2).*

3. Dynamic Programming

A dynamic programming algorithm based on the notion of relative value iteration (Abounadi et al., 2001) is now developed. Let P_{μ^k} denote the transition probability matrix associated with the policy μ^k. Then if the Markov chain of the policy is irreducible and positive recurrent, its invariant measure (steady-state probabilities) can be determined by solving the following linear system of equations (also called the invariance equations):

$$\Pi^k P_{\mu^k} = \Pi^k; \qquad \sum_{i \in \mathcal{S}} \Pi^k(i) = 1, \tag{11}$$

where Π^k denotes the row vector of the steady state probabilities. Using these probabilities, one can compute the first moment of the immediate reward for the policy μ^k via the following equation:

$$\rho_{\mu^k} = \sum_{i \in \mathcal{S}} \Pi^k(i) \left[\sum_{j \in \mathcal{S}} p(i,a,j) r(i,a,j) \right]. \tag{12}$$

Steps in Value Iteration: Select any strictly positive value for ϵ. An ϵ-optimal policy obtained from this algorithm will be denoted as μ_ϵ. The algorithm is initialized with an arbitrary values for the function $Q^1 : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ and an arbitrary value for the scalar $\rho^1 \in \mathbb{R}$. Set $k = 1$. Chose any state-action pair in the system to be a distinguished state-action pair, and call it (i^*, a^*).

Step 1. Compute $J^k(i) = \max_{a \in \mathcal{A}(i)} Q^k(i,a)$ for all $i \in \mathcal{S}$.

Step 2. Update the Q-factors for all (i,a) pairs as follows:

$$Q^{k+1}(i,a) = (1-\alpha(k))Q^k(i,a) + \alpha(k)\mathcal{E}^k(i,a), \text{ where } \mathcal{E}^k(i,a) \text{ is given by}$$

$$\sum_{j \in \mathcal{S}} p(i,a,j) \left[r(i,a,j) - \theta(r(i,a,j) - \rho^k)^2 + \max_{b \in \mathcal{A}(j)} Q^k(j,b) - Q(i^*,a^*) \right]. \tag{13}$$

Step 3. Let $\mu^{k+1}(i) \in \arg\max_{a \in \mathcal{A}(i)} Q^{k+1}(i,a)$. Compute the first moment of the long-run reward $\rho_{\mu^{k+1}}$ of μ^{k+1} via Equations (11)—(12). Then update ρ^k as follows:

$$\rho^{k+1} = (1-\beta(k))\rho^k + \beta(k)\rho_{\mu^{k+1}}. \tag{14}$$

Step 4. Compute $J^{k+1}(i) = \max_{a \in \mathcal{A}(i)} Q^k(i,a)$ for all i. If $||J^{k+1} - J^k|| < \epsilon$, stop and set $\mu_\epsilon(i) = \mu^{k+1}(i)$ for every $i \in \mathcal{S}$. Otherwise, increment k by 1, and return to Step 2.

To analyze the convergence of this algorithm, the iterate Q^k is first considered, which can be written as:

$$Q^{k+1}(i,a) = (1-\alpha(k))Q^k(i,a) + \alpha(k)F_{i,a}(Q^k,\rho^k) \text{ where } F_{i,a}(Q^k,\rho^k) =$$

$$\sum_{j \in \mathcal{S}} p(i,a,j) \left[r(i,a,j) - \theta(r(i,a,j) - \rho^k)^2 + \max_{b \in \mathcal{A}(j)} Q^k(j,b) - Q^k(i^*,a^*) \right]. \tag{15}$$

The following definition is needed: For all (i,a) pairs:

$$f_{i,a}(Q^k,\rho^k) = F_{i,a}(Q^k,\rho^k) - Q^k(i,a). \tag{16}$$

Now, consider the slower iterate, ρ^k. If

$$\psi^k = \sum_{i \in \mathcal{S}} \Pi^k(i) \left[\sum_{j \in \mathcal{S}} p(i,a,j) r(i,a,j) \right], \tag{17}$$

then the update of ρ^k can be expressed as follows:

$$\rho^{k+1} = \rho^k \beta(k) + \beta(k)(\psi^k - \rho^k).$$

Further define

$$g(Q^k,\rho^k) = \psi^k - \rho^k. \tag{18}$$

Let $\acute{\rho}$ and $\acute{Q}$ denote the time-interpolated continuous-time processes underlying the iterates ρ^k and Q^k respectively (Kushner and Yin, 2003). Now, using the definitions of $f(,.,)$ and $g(,.,)$ and setting $m = (i,a)$, where $m = 1, 2, \ldots, n$, where n denotes the number of state-action pairs in the system, one can express the (synchronous) updating of the iterates on two timescales as:

$$Q^{k+1}(m) = Q^k(m) + \alpha(k) f_m(Q^k, \rho^k) \text{ for } m = 1, \ldots.., n \tag{19}$$

$$\text{and } \rho^{k+1} = \rho^k + \beta(k) g(Q^k, \rho^k). \tag{20}$$

The idea here is to use an ODE associated with any fixed value of ρ. For any $\rho \in \mathbb{R}$, consider the ODE:

$$\frac{d\acute{Q}}{dt} = f(\acute{Q}, \acute{\rho}). \tag{21}$$

From the definition of the function $f_{i,a}(.)$ in Equation (16), for any $\rho \in \mathbb{R}$, any critical point of the ODE above can be written as:

$$Q^{\infty}_{\rho}(i,a) = F_{i,a}(Q^{\infty}_{\rho}, \rho), \text{ for all } (i,a)\text{-pairs} \tag{22}$$

where Q^{∞}_{ρ} denotes the critical point and $F_{i,a}(.,.)$ is as defined in (15).

A useful condition (Assumption 3.1) that will be necessary in the convergence analysis for showing Lipschitz continuity of a certain function will now be derived. Some additional work is needed to derive this condition:

Differentiate both sides of Equation (22) with respect to ρ. After some simple algebra, one obtains:

$$M \cdot \frac{\partial Q^{\infty}_{\rho}}{\partial \rho} = A \cdot [1, \rho]^T \tag{23}$$

where M is an $n \times n$ matrix, A is an $n \times 2$ matrix, and $\frac{\partial Q^{\infty}_{\rho}}{\partial \rho}$ is a column vector of the partial derivatives $\frac{\partial Q^{\infty}_{\rho}(.,.)}{\partial \rho}$. The terms in $F_{i,a}(Q^{\rho}_{\infty}, \rho)$ that contain the ρ terms will be (see Equation (15)):

$$2\theta\rho \sum_{j \in \mathcal{S}} p(i,a,j) r(i,a,j) - \theta\rho^2.$$

Therefore, the mth row of A will equal (remember $m \equiv (i,a)$) for $m = 1, \ldots, n$:

$$\left[2\theta \sum_{j \in \mathcal{S}} p(i,a,j) r(i,a,j), -2\theta \right]. \tag{24}$$

The condition needed for Lipschitz continuity is as follows:

Assumption 3.1 The matrix M defined via Equations (23) and (24) is invertible.

Some standard assumptions on step sizes that are usually made within stochastic approximation over two time-scales will also be needed:

Assumption 3.2

$$\lim_{k\to\infty} \sum_{l=1}^{k} \alpha(l) = \infty; \quad \lim_{k\to\infty} \sum_{l=1}^{k} \beta(l) = \infty; \quad \lim_{k\to\infty} \frac{\beta(k)}{\alpha(k)} = 0.$$

An example for step sizes that satisfy the rules above is:

$$\alpha(k) = \frac{\log(k+1)}{k+1}; \qquad \beta(k) = \frac{c_1}{c_2 + k} \text{ for some positive scalars } c_1 \text{ and } c_2. \tag{25}$$

The following is the main convergence result for the value iteration algorithm.

Theorem 3.3: *Let $\mu^k(i) \in \arg\max_a Q^k(i,a)$ for every $i \in \mathcal{S}$. Then under Assumptions 2.9, 3.1, and 3.2, $\mu^k \to \mu_*$.*

Since the proof is rather long, and so as to not distract the reader from the general applicability of the result, the proof is relegated to Appendix B.

4. Reinforcement Learning

A reinforcement learning algorithm, based on Q-Learning, for the case in which transition probabilities are not available is now proposed. It will be assumed that a randomized stationary policy will be applied to the system. The updating in RL is asynchronous because one goes randomly from one state to another. To formulate a mathematical model for this scenario, consider a stochastic process $\{Y(l)\}$ that takes values from the set $\{1, 2, \ldots, n\}$, where n denotes the number of state-action pairs; $Y(l)$ will denote the state-action pair visited in the lth iteration of the algorithm. For every state-action pair (i,a), define a counter $\nu^k(i,a)$ as follows:

$$\nu^k(i,a) = \sum_{l=1}^{k} I\{Y(l) = (i,a)\},$$

where $I\{.\}$ denotes the indicator function. Thus the counter represents the number of times state-action pair (i,a) has been visited until and including the kth iteration. Steps in the RL algorithm are as follows:

Step 1. Set k, the number of state transitions, to 1. Initialize the Q-factors: set for all (l,u) where $l \in \mathcal{S}$ and $u \in \mathcal{A}(l)$: $Q^k(l,u) = 0$. Set $\bar{\gamma}(l) = 1$ for all $i \in \mathcal{S}$, where $\bar{\gamma}(l)$ denotes the number of times state l has been visited thus far. The algorithm will be run for $k_{\max}$ iterations, where $k_{\max}$ is chosen to be a sufficiently large number. Start system simulation at any arbitrary state. Chose a value for C in the interval $(0,1)$. Chose any state-action pair in the system to be a distinguished state-action pair, and call it (i^*, a^*).

Step 2. Let the current state be i. Set $\bar{\epsilon} = C/\bar{\gamma}(i)$. The so-called greedy action is one that maximizes the Q-factor (i.e., belongs to the set $\arg\max_{i \in \mathcal{A}(i)} Q(i,a)$). Choose a greedy action with a probability of $1 - \frac{\bar{\epsilon}}{|\mathcal{A}(i)-1|}$ and any one of the remaining $|\mathcal{A}(i) - 1|$ actions with a probability of $\frac{\bar{\epsilon}}{|\mathcal{A}(i)-1|}$.

Step 3. Simulate action a. Let the next state be j. Let $r(i,a,j)$ be the immediate reward earned in the transition to state j from state i under the influence of action a. The quantity $r(i,a,j)$ will be determined by the simulator.

Step 4. Update $Q^k(i,a)$ using the following equation:

$$Q^{k+1}(i,a) = Q^k(i,a) + \alpha(k) \times [r(i,a,j) - \theta(r(i,a,j) - \rho^k)^2 + \max_{b \in \mathcal{A}(j)} Q^k(j,b)$$
$$- Q^k(i^*, a^*) - Q^k(i,a)].$$

Step 5. If a greedy action was chosen in Step 3, update ρ^k as follows:

$$\rho^{k+1} = \rho^k + \beta(k)\left[r(i,a,j) - \rho^k\right].\text{Otherwise, set } \rho^{k+1} = \rho^k.$$

Step 6. Increment k by 1 and $\bar{\gamma}(i)$ by 1. If $k < k_{\max}$, set $i \leftarrow j$, and go to Step 2. Otherwise, go to Step 7.

Step 7. For each $l \in \mathcal{S}$, select $\mu^k(l) \in \arg\max_{b \in \mathcal{A}(l)} Q^k(l,b)$. The policy (solution) generated by the algorithm is μ^k. Stop.

To analyze the convergence of the algorithm, the format in (19) and (20) will be reused after noting that RL produces *asynchronous* updating. The choice of step sizes needs to satisfy some additional conditions needed for asynchronous convergence. These are:

Assumption 4.1 Let $\gamma(l)$ denote the step-size in the lth iteration. Then,

(i) $\lim_{k\to\infty} \sum_{l=1}^{k} (\gamma(l))^2 < \infty$.
(ii) $\gamma(l+1) \le \gamma(l)$ eventually.
(iii) for $x \in (0,1)$, $\sup_k \gamma([xk])/\gamma(k) < \infty$,
(iv) for $x \in (0,1)$, $\left(\sum_{l=0}^{[xk]} \gamma(l)\right) / \left(\sum_{l=0}^{k} \gamma(l)\right) \to 1$, where $[\Lambda]$ denotes the integer part of Λ.

Assumption 4.2 There exists a $\Gamma > 0$ such that

(i) $\liminf_{k\to\infty} \frac{\nu^k(i,a)}{k+1} \ge \Gamma$ almost surely for every (i,a)-pair.
(ii) If $N(k,x) = \min\left\{q \ge k : \sum_{l=l}^{q} \gamma(l) \ge x\right\}$ for any $x > 0$, then the limit $\lim_{k\to\infty} \frac{\sum_{l=\nu^k(i,a)}^{\nu^{N(k,x)}(i,a)} \gamma(l)}{\sum_{l=\nu^k(j,b)}^{\nu^{N(k,x)}(j,b)} \gamma(l)}$ exists almost surely for all i,j,a,b.

Fortunately, it is not hard to identify step sizes that satisfy both the Assumption 4.1 and part (ii) of Assumption 4.2; step sizes in (25) form one example. Part (i) of Assumption 4.2 holds when all state-action pairs are tried infinitely often. The exploration strategy chosen in the RL algorithm can be shown to ensure this (see Singh et al. (2000)).

Theorem 4.3: *Let $\mu^k(i) \in \arg\max_{a\in\mathcal{S}} Q^k(i,a)$ for every $i \in \mathcal{S}$. If Assumptions 2.9, 3.1, and 3.1 hold, and both $\alpha(k)$ and $\beta(k)$ satisfy Assumptions 4.1 and 4.2, then μ^k will converge to μ_* with probability 1 .*

The proof is similar to that of Theorem 3.3 with some synthetic differences, and is hence presented in Appendix C.

5. Numerical Results

This section contains results from numerical tests with the dynamic programming and RL algorithms. Results from a small MDP with two states are presented in subsection 5.1. Subsection 5.2 discusses results from a bigger problem from the domain of preventive maintenance.

5.1 *Two-state MDP*

Consider a two-state-two-action problem. Let P_a and R_a denote the transition probability and reward matrices for choosing action a in every state respectively;

then, $P_a(i,j) \equiv p(i,a,j)$ and $R_a(i,j) \equiv r(i,a,j)$. Further $\theta = 0.15$.

$$P_1 = \begin{bmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{bmatrix}; P_2 = \begin{bmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{bmatrix}; R_1 = \begin{bmatrix} 6.0 & -5 \\ 7.0 & 12 \end{bmatrix}; R_2 = \begin{bmatrix} 5.0 & 68 \\ -2 & 12 \end{bmatrix}.$$

The optimal policy is $(1,2)$, and the value iteration algorithm converges to it for $\epsilon = 0.1$ in 18 iterations. The optimal values for this problem, determined via exhaustive evaluation, are: $\rho_* = 8.6250$ and $\phi_* = 3.9323$. Table 1 shows the convergence in the norm and also shows that the value iteration algorithm converges to values close to ρ_* and ϕ_*.

Table 1. Calculations in value iteration: Set $(i^*, a^*) = (1,1)$. The algorithm converged in 18 iterations. At the end, $Q^{18}(i^*, a^*) = 3.8737 \simeq 3.9323 = \phi_*$.

k	$Q(1,1)$	$Q(1,2)$	$Q(2,1)$	$Q(2,2)$	ρ^k
1	0	0	0	0	0
2	-1.7636	-49.1382	-4.7191	-7.1186	4.6536
3	-2.4089	-49.1903	1.4850	-1.439	5.5898
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
18	3.8737	-39.9378	29.3185	38.1220	8.6250

The RL algorithm was run for 30,000 iterations within a Markov chain simulator. It needs to be under-scored that unlike one iteration in the dynamic programming algorithm, where one may have to solve linear equations, one iteration in RL is associated with one transition of the Markov chain and takes very little computational time. On an Intel Pentium Processor with a speed of 2.66 GHz and a 64-bit operating system, the RL and the DP algorithms took no more than 10 seconds. All the codes were written in MATLAB. Table 2 shows the Q-values obtained from the RL algorithm run with full exploration. As is clear, the algorithm converges to the optimal policy. The value function to which it converges is: $(3.1486, 35.0089)$, which is reasonably close to the value function to which the value iteration algorithm converged: $(3.8737, 38.1220)$. Further, when the algorithm is terminated, ρ^k (=6.5081) converged to the vicinity of its optimal value (=8.6250). After the optimal policy was determined, ρ^k was estimated in a fresh simulation using the optimal policy, and it converged to $8.633 \simeq \rho_*$.

Table 2. Calculations in the RL algorithm: Set $(i^*, a^*) = (1,1)$. At the end, the policy is (1,2), which is an optimal policy. Also, $Q(i^*, a^*) = 3.1486 \simeq 3.9323 = \phi_*$.

k	$Q^k(1,1)$	$Q^k(1,2)$	$Q^k(2,1)$	$Q^k(2,2)$	ρ^k
1	0	0	0	0	0
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
30,000	3.1486	-12.4945	31.7979	35.0089	6.5081

5.2 *Preventive Maintenance*

In this subsection, the goal is to test the algorithms developed on a problem with a relatively larger state-space. A problem from the domain of preventive maintenance of production lines (Gosavi, 2006), where input parameters were derived from an industrial data set, is selected.

Consider a production line that is prone to failures. When it fails, it is repaired by the start of the next day. Failures cause severe disruptions and are costly. A commonly used strategy to reduce the occurrence of failures is preventive maintenance in which the line is maintained periodically. In general, in complex systems such as a production line, the probability of failure increases as the time elapsed since last repair of maintenance increases. We will assume that after each maintenance or repair, the system is set back to its original state. In such a setting, the state of the system can be defined by the number of days since last repair or maintenance. Each repair will be assumed to cost C_r dollars while each maintenance will be assumed to cost C_m dollars. We will further assume that when the system is maintained, it is down for an entire day. There are two actions from which the manager must choose at the start of each day: *Continue Production, Maintain System.*

The transition probabilities will be defined as follows using the data in Gosavi (2006). The state space, $\mathcal{S}$ will be $\{0, 1, 2, \ldots, 30\}$.

When action a equals "continue production," for $i = 0, 1, \ldots, 29$: $p(i, a, i + 1) = 0.99 * \Lambda^i; p(i, a, 0) = 1 - 0.99 * \Lambda^i$. For $i = 30$, $p(i, a, 0) = 1$. Also, $r(i, a, 0) = -C_r$ for all $i \in \mathcal{S}$. All other values of $r(., ., .)$ and $p(., ., .)$ will equal zero.

When action a equals "maintain," for all $i \in \mathcal{S}$, $p(i, a, 0) = 1$ and $r(i, a, 0) = -C_m$. All other values of $p(., ., .)$ and $r(., ., .)$ will equal zero.

Experiments were performed with differing values for the inputs of C_r, C_m, θ and Λ to generate 8 different cases. For each case, an exhaustive evaluation was performed, via Equation (2), of all deterministic policies to determine the optimal policy. Table 3 shows the results obtained from these numerical experiments. The table provides, k^{ϵ}_{DP}, the number of iterations needed by the DP algorithm to generate the ϵ-optimal policy, in addition to the percentage deviation from optimality of the RL algorithm. The percentage deviation of the RL algorithm is measured as follows:

$$\chi(RL) \equiv \left| \frac{\phi_{RL} - \phi^*}{\phi^*} \right| \times 100,$$

where ϕ_{RL} denotes the objective function value of the policy produced by the RL algorithm and ϕ^* denotes the same for the optimal policy. The objective function value for each policy (RL and optimal) was evaluated from using Equation (2). Note that since the calculations are performed in terms of costs, while the definition of ϕ is in terms of rewards, the value of $-\phi^*$ is shown in Table 3.

The optimal policy for these problems turns out to be of the threshold kind, i.e., the optimal policy recommends "continue production" until a certain value of i and then recommends "maintaining" from state i^* onwards. The value of i^* for the optimal policy in each case is also shown in Table 3. As is clear, the RL algorithm produced optimal or near-optimal solutions in every case tested. The DP algorithm generated the optimal solution in every case. The computational time did not exceed 2 minutes in any of these problems using the same computer specified above.

6. Conclusions

While the variance-penalized MDP in risk-averse Markov control was developed a while back (Filar et al., 1989), the associated Bellman equation is a new development. The theory proposed here for Bellman equations (in Propositions 2.6 and

Table 3. Numerical results with the preventive maintenance case study: Set $\epsilon = 0.001$ in DP. The RL algorithm was run for 30,000 iterations.

Case	C_m	C_r	Λ	θ	i^*	k^{ϵ}_{DP}	$\chi(RL)$	$-\phi^*$
1	3	4	0.95	0.1	8	29	5.22%	0.8312
2	2	4	0.95	0.3	4	35	8.07%	0.9856
3	3	4	0.95	0.3	7	34	0.43%	1.2300
4	3	4	0.97	0.5	9	44	3.59%	1.3589
5	3	4	0.94	0.5	6	34	0.00%	1.7239
6	4	5	0.94	0.5	7	36	0.04%	2.5480
7	4	5	0.96	0.5	9	40	2.48%	2.2178
8	4	6	0.96	0.5	5	47	0.27%	2.7536

2.8) for the problem is a new contribution; see however Sato and Kobayashi (2001), who presented a Q-factor version of the optimality equation, albeit without any analysis. The analysis for showing the optimality of the proposed Bellman optimality equation first required developing the theory for an associated Bellman equation for a given policy. The second contribution here was in formulating dynamic programming and reinforcement learning algorithms using the Bellman optimality equation for solving the problem. Algorithms presented in this paper can be potentially useful in revenue management scenarios where a risk-averse manager seeks to minimize variability in revenues while maximizing average revenues. Examples of such problems abound in pricing of airline tickets (revenue management), in queuing control for wireless communications (where variance in rate of dropped calls is of concern), and financial transactions (portfolio allocation). In the operation of complex systems and System of Systems (SoS) theory, these algorithms can be used for dynamic decision-making in numerous scenarios, including where a government has to allocate capital to infrastructure projects (Mostafavi et al., 2012) and where a centralized agent needs to control a team of robots (Dudek et al., 1996). In such systems, the state of all the sub-systems (infrastructure projects or robots) can be combined to form an aggregate state for the MDP in which the decision-maker is the supervising system which is responsible for selecting actions that influence the SoS as a whole.

A number of directions for future research can be envisioned on the basis of the work presented here. A few directions are enumerated below.

(1) Further refinements of the dynamic programming algorithm should consider techniques that could potentially reduce the computational burden incurred in computing the steady-state probabilities in every iteration. While it is clear that if the policy does not change in an iteration, these probabilities need not be recomputed, our current algorithm does not ensure that a policy is not revisited after a few iterations. Other alternatives worth consideration are Markov Chain Monte Carlo schemes for the slower time scale that would converge to ρ_* without the use of the invariance equations.

(2) An important future direction that remains is to establish that the Bellman optimality equation proposed here is guaranteed to have a solution. In the classical risk-neutral version of the MDP, this can be done using the notion of Blackwell optimality, which may not have an obvious counterpart here due to the difficulties with using a discount factor (see Filar et al. (1989)).

(3) Another important theoretical aspect of the problem that should be considered is the notion of denumerable state spaces and an analysis when the regularity condition used here fails to hold for the underlying Markov chains.

(4) Assumption 2.7 is evidently a rather strong condition. It will be very useful to establish convergence without it. Population-based stochastic search algorithms (Chang et al., 2007) may not require this condition, and are potentially a new and exciting avenue for problems of this nature that defy the classical framework of Markov decision processes.
(5) An interesting extension of this model would be to interacting systems in an SoS (Wernz and Deshmukh, 2010) where the supervising agent may be interested in risk-averse decision making while the agents at a lower level may be interested in risk-neutral behavior.

References

J. Abounadi, D.P. Bertsekas, and V.S. Borkar. Learning algorithms for Markov decision processes with average cost. *SIAM Journal of Control and Optimization*, 40:681–698, 2001.

C. Barz and K.H. Waldmann. Risk-sensitive capacity control in revenue management. *Mathematical Methods of Operations Research*, 65:565–579, 2007.

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

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

K. Boda and J. Filar. Time consistent dynamic risk measures. *Mathematical Methods of Operations Research*, 63:169–186, 2005.

V. S. Borkar. *Stochastic Approximation: A Dynamical Systems Viewpoint*. Hindustan Book Agency and Cambridge University Press (jointly), Delhi, India and Cambridge, UK, 2008.

M. Bouakiz and Y. Kebir. Target-level criterion in Markov decision processes. *Journal of Optimization Theory and Applications*, 86:1–15, 1995.

R. Cavazos-Cadena and E. Fernandez-Gaucherand. Controlled Markov chains with risk-sensitive criteria: Average cost, optimality equations, and optimal solutions. *Mathematical Methods of Operations Research*, 43:121–139, 1999.

H.S. Chang, M.C. Fu, J. Hu, and S. I. Marcus. *Simulation-based Algorithms for Markov Decision Processes*. Springer, 2007.

G. Dudek, M.R. Jenkin, E. Milios, and D. Wilkes. A taxonomy for multi-agent robotics. *Autonomous Robots*, 3(4):375–397, 1996.

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

J. Filar, D. Krass, and K. Ross. Percentile performance criteria for limiting average Markov decision processes. *IEEE Transactions on Automatic Control*, 40:2–10, 1995.

Peter Geibel and Fritz Wysotzki. Risk-sensitive reinforcement learning applied to chance constrained control. *Journal of Artificial Intelligence Research*, 24, 2005.

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

A. Gosavi. Adaptive critics for airline revenue management. In *The 18th Annual Conference of the Production and Operations Management Society (POMS)*. POMS, May 2007.

A. Gosavi. Reinforcement learning for model building and variance-penalized control. In *2009 Winter Simulation Conference (WSC'09)*, Austin (TX), USA, 2009.

A. Gosavi. Target-sensitive control of Markov and semi-Markov processes. *International Journal of Control, Automation, and Systems*, 9(5):1–11, 2011.

A. Gosavi and S.P. Meyn. Value iteration on two time-scales for variance-penalized

Markov control. In A. Johnson and J. Miller, editors, *Proceedings of the 2010 Industrial Engineering Research Conference*, 2010.

A. Gosavi, S.L. Murray, V.M. Tirumalasetty, and S. Shewade. A Budget-Sensitive Approach to Scheduling Maintenance in a Total Productive Maintenance (TPM) Program. *Engineering Management Journal*, 23(3):46–56, 2011.

C.M. Grinstead and J.L. Snell. *Introduction to Probability.* American Mathematical Society, Providence, RI, USA, 1997.

X. P. Guo. Constrained optimization for average cost continuous-time Markov decision processes. *IEEE Transactions on Automatic Control*, 52(6):1139–1143, 2007.

M. Heger. Consideration of risk in reinforcement learning. In *Proceedings of the 11th International Conference on Machine Learning*, pages 105–111. Morgan Kaufmann, 1994.

R. Howard and J. Matheson. Risk-sensitive Markov decision processes. *Management Science*, 18(7):356–369, 1972.

Y. Huang, X. Guo, and Z. Li. Minimum risk probability for finite horizon semi-Markov decision process. *Journal of Mathematical Analysis and Applications*, 402(1):378–391, 2013.

H.J. Kushner and G. Yin. *Stochastic Approximation and Recurstive Algorithms and Applications.* Springer Verlag, New York, second edition, 2003.

A. E. B. Lim and X.Y. Zhou. Risk-sensitive control with HARA utility. *IEEE Transactions on Automatic Control*, 46(4):563–578, 2001.

M. Littman and C. Szepesvari. A generalized reinforcement-learning model: Convergence and applications. In *Proceedings of the 13th International Conference on Machine Learning*, pages 310–318. Morgan Kaufmann, 1996.

O. Mihatsch and R. Neuneier. Risk-sensitive reinforcement learning. *Machine Learning*, 49(2-3):267–290, November 2002.

D. Mitra and Q. Wang. Stochastic traffic engineering for demand uncertainty and risk-aware network revenue management. *IEEE/ACM Transactions on Networking*, 13(2):221–232, 2005.

A. Mostafavi, D. Abraham, S. Noureldin, G. Pankow, J. Novak, R. Walker, K. Hall, and B. George. Risk-based protocol for the inspection of transportation construction projects undertaken by state departments of transportation. *Journal of Construction Engineering and Management, ASCE*, 139(8):977–986, 2012.

Y. Niv, J. Edlund, P. Dayan, and J. O'Doherty. Neural prediction errors reveal a risk-sensitive reinforcement-learning process in the human brain. *Journal of Neuroscience*, 32(2):551–562, 2012.

B. T. Polyak. *Introduction to Optimization.* Optimization Software, NY, 1987.

S. M. Ross. *Introduction to Probability Models.* Academic Press, San Diego, CA, USA, 1997.

M. Sakaguchi and Y. Ohtsubo. Optimal threshold probability and expectation in semi-Markov decision processes. *Applied Mathematics and Computation*, 216: 2947–2958, 2010.

M. Sato and S. Kobayashi. Average-reward reinforcement learning for variance penalized Markov decision problems. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 473–480, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

S. Singh, T. Jaakkola, M. Littman, and C. Szepesvari. Convergence results for single-step on-policy reinforcement-learning algorithms. *Machine Learning*, 39: 287–308, 2000.

M. Sobel. The variance of discounted Markov decision processes. *Journal of Applied Probability*, 19:794–802, 1982.

R. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction.* The MIT Press, Cambridge, MA, USA, 1998.

C.J. Watkins. *Learning from Delayed Rewards.* PhD thesis, Kings College, Cambridge, England, May 1989.

C. Wernz and A. Deshmukh. Multi-time-scale decision making for strategic agent interactions. In A. Johnson and J. Miller, editors, *Proceedings of the 2010 Industrial Engineering Research Conference*, Cancun, Mexico, 2010. IIE.

D. White. Minimizing a threshold probability in discounted Markov decision processes. *Journal of Mathematical Analysis and Applications*, 173:634–646, 1993.

C. Wu and Y. Lin. Minimizing risk models in Markov decision processes with policies depending on target values. *Journal of Mathematical Analysis and Applications*, 231:47–67, 1999.

Appendix:

Appendix A Proof of Proposition 2.6

Proof: Equation (3) can be written in vector form as:

$$\phi e + h = L_\mu(h), \tag{A1}$$

where e is an $|\mathcal{S}|$-dimensional (column) vector whose every element equals 1. It will be first proved that: $L_\mu^k(h) = k\phi e + h$, i.e., for $i = 1, 2, \ldots, |\mathcal{S}|$,

$$L_\mu^k h(i) = k\phi + h(i). \tag{A2}$$

From Equation (A1), the above is true when $k = 1$. Assuming it is true when $k = m$, one has that $L_\mu^m(h) = m\phi e + h$. Using the transformation L_μ on both sides of this equation, the induction can be completed as follows:

$$\begin{aligned} L_\mu\left(L_\mu^m(h)\right) &= L_\mu(m\phi e + h) \\ &= m\phi e + L_\mu(h) = (m+1)\phi e + h \text{ (using Equation (A1))}. \end{aligned}$$

Using Lemma 2.4, one has that for all i:

$$L_\mu^k(h(i)) = E_\mu\left[h(x_{k+1}) + \sum_{s=1}^{k}\left[\bar{r}(x_s, \mu(x_s)) - \theta\bar{v}_\mu(x_s)\right]\middle| x_1 = i\right],$$

where $|h(x_{k+1})| < \infty$.

Using the above and Equation (A2), one has that:

$$\mathsf{E}_\mu\left[h(x_{k+1}) + \sum_{s=1}^{k}\left[\bar{r}(x_s, \mu(x_s)) - \theta\bar{v}_\mu(x_s)\right]\middle| x_1 = i\right] = k\phi + h(i).$$

Therefore,

$$\frac{\mathsf{E}_\mu[h(x_{k+1})]}{k} + \frac{\mathsf{E}_\mu\left[\sum_{s=1}^{k}\left[\bar{r}(x_s, \mu(x_s)) - \theta\bar{v}_\mu(x_s)\right]\middle| x_1 = i\right]}{k} = \phi + \frac{h(i)}{k}.$$

Taking limits as $k \to \infty$,

$$\lim_{k\to\infty}\frac{1}{k}\mathsf{E}_\mu\left[\sum_{s=1}^{k}\left[\bar{r}(x_s, \mu(x_s)) - \theta\bar{v}_\mu(x_s)\right]\middle| x_1 = i\right] = \phi.$$

Then from Definitions 2.1 and 2.2, the above implies that $\phi_\mu = \phi$. □

Appendix B Proof of Theorem 3.3

The overall roadmap for the proof of Theorem 3.3 is as follows. If the slower iterate (ρ^k) converges to its optimal solution, one can show by invoking a two time-scale result (Borkar, 2008, Theor.2; Chap. 6) that under some conditions, namely the boundedness of the iterates and Lipschitz continuity of some functions, the faster iterates $(Q^k(.,.))$ will also converge to the desired solution. That both iterates will remain bounded will be shown in Lemma B.1. Via Lemma B.2, the Lipschitz continuity of the relevant functions will be established. Then, the slower iterate, ρ^k, will be shown to converge to ρ_*, via Lemma B.4, from which, using the two time-scale result, convergence to an optimal policy is immediate.

From the definition of ψ^k in (17), one has that there exists a bound $\overline{\psi} < \infty$ on the sequence $\{\psi^k\}_{k=1}^{\infty}$:

$$\sup_k |\psi^k| \leq \overline{\psi}. \tag{B1}$$

This bound implies that the iterates will remain bounded.

Lemma B.1: *The sequence $\{Q^k, \rho^k\}_{k=1}^{\infty}$ remains bounded under Assumption 3.2.*

Proof: Induction will be used to show that $\bar{\rho} = \max(|\rho^1|, \overline{\psi}) > 0$ provides a bound on the sequence $\{\rho^k\}_{k=1}^{\infty}$. The induction step follows from Equation (14): If $|\rho^k| \leq \bar{\rho}$ then,

$$|\rho^{k+1}| < (1 - \beta(k))|\rho^k| + \beta(k)\overline{\psi} < (1 - \beta(k))\bar{\rho} + \beta(k)\bar{\rho} = \bar{\rho}.$$

To establish boundedness of $\{Q^k\}_{k=1}^{\infty}$, the ODE method will be applied. Define for any $\nu > 0$,

$$f_{i,a}^{\nu}(Q^k, \rho) = \frac{f_{i,a}(\nu Q^k, \rho)}{\nu} \quad \text{for every } (i, a),$$

and consider the limiting vector field $f_{i,a}^{\infty} \equiv \lim_{\nu\to\infty} f_{i,a}^{\nu}$. Using (16), one can conclude that this is independent of ρ, and for any (i, a) is given by,

$$f_{i,a}^{\infty}(Q^k, \rho) = f_{i,a}^{\infty}(Q^k) = \sum_{j\in\mathcal{S}} p(i, a, j) \left[\max_{b\in\mathcal{A}(j)} Q^k(j, b) - Q^k(i^*, a^*)\right] - Q^k(i, a).$$

For any fixed ρ, the underlying ODE is given by: $\frac{d\acute{Q}}{dt} = f^{\infty}(\acute{Q})$. A special case of the result in Theorem 3.4 in Abounadi et al. (2001) (for the classical relative value iteration algorithm) implies that the origin is the globally asymptotically stable equilibrium point for this ODE. Then, Borkar (2008, Theor. 7; Chap 3) implies that $\{Q^k\}_{k=1}^{\infty}$ is a bounded sequence. □

Lemma B.2: *The function $f(Q^k, \rho^k)$ is Lipschitz in Q^k and $g(Q^k, \rho^k)$ is Lipschitz in ρ^k. Further, under Assumption 3.1, the critical point of the ODE in Equation (21), $F(Q_{\infty}^{\rho}, \rho)$, will be Lipschitz in ρ.*

Proof: Since $f(Q^k, \rho^k)$ has bounded derivatives with respect to Q^k, it is Lipshitz in Q^k. Similarly, $g(Q^k, \rho^k)$ is Lipschitz in ρ^k.

Since M is invertible under Assumption 3.1, using Equation (23) one has that:

$$\frac{\partial Q}{\partial \rho} = M^{-1} \cdot A \cdot [1, \rho]^T,$$

which implies using the definition of A in Equation (24) that the derivatives of

$F(Q_\infty^\rho, \rho)$ with respect to ρ are bounded on the domain $[-\bar{\rho}, \bar{\rho}]$, proving the Lipschitz continuity in ρ. □

The following limits (see Polyak (1987)) will be useful in showing that the slower iterate converges.

Lemma B.3: *For a non-negative sequence* $\{\beta(i)\}_{i=1}^\infty$ *satisfying Assumption 3.2,*

$$\lim_{k\to\infty} \prod_{i=1}^{k} (1 - \beta(i)) = 0. \tag{B2}$$

Moreover, for any sequence $\{\delta^k\}_{k=1}^\infty$ *satisfying* $\lim_{k\to\infty} \delta^k = 0$*, one has*

$$\lim_{k\to\infty} \sum_{i=1}^{k} \left(\prod_{j=i+1}^{k} (1 - \beta(j)) \right) \beta(i)\delta^i = 0. \tag{B3}$$

The following establishes that ρ^k will converge to its optimal solution.

Lemma B.4: $\rho^k \to \rho_*$ *under Assumptions 3.2 and 3.1.*

Proof: (Lemma B.4) This proof will be similar in spirit to that of Theorem 4.5 of Abounadi et al. (2001). A sequence $\{\Delta^k\}_{k=1}^\infty$ where $\Delta^k = \rho^k - \rho_*$ will be set up, so that the goal becomes to show that $\Delta^k \to 0$. The roadmap for the proof is as follows. First define another sequence $\{\delta^k\}_{k=1}^\infty$ and express Δ^k in terms of δ^k. Then, construct upper and lower bounds on the partial derivative of $g(Q^k, \rho^k)$ with respect to ρ^k. These bounds, the sequence $\{\delta^k\}_{k=1}^\infty$, and Lemma B.3 will together be exploited to show that $\Delta^k \to 0$.

$$\text{Let } \delta^k = g(Q^k, \rho^k) - g(Q_\infty^{\rho^k}, \rho^k). \tag{B4}$$

Under Assumption 3.2, Lemmas B.1 and B.2 allow one to invoke the two time-scale convergence result, which implies that for a fixed value of ρ, Q^k will converge to the fixed point Q_∞^ρ, satisfying (22) i.e., $\delta^k \to 0$.

Δ^k will now be expressed in terms of $g(.,.)$ and δ^k. From the definition of δ^k above in Equation (B4) and the fact that the update of ρ^k in Step 3 of the algorithm can be expressed as follows:

$$\rho^{k+1} \leftarrow \rho^k + \beta(k)(g(Q^k, \rho^k)), \tag{B5}$$

one has that:

$$\Delta^{k+1} = \Delta^k + \beta(k)g(Q_\infty^{\rho^k}, \rho^k) + \beta(k)\delta^k. \tag{B6}$$

From Equation (18), $\frac{\partial g(Q^k,\rho^k)}{\partial \rho^k} = -1 < 0$ since ψ^k is not a function of ρ^k. Hence, there exist negative, upper and lower bounds on the derivative, i.e., there exist $C_1, C_2 \in \mathbb{R}$ where $0 < C_1 \leq C_2$ such that:

$$-C_2(\rho_1 - \rho_2) \leq g(Q_\infty^{\rho_1}, \rho_1) - g(Q_\infty^{\rho_2}, \rho_2) \leq -C_1(\rho_1 - \rho_2) \tag{B7}$$

for any $\rho_1, \rho_2 \in \mathbb{R}$ if $\rho_1 > \rho_2$. If $\rho_2 > \rho_1$, one has the following inequality:

$$-C_2(\rho_1 - \rho_2) \geq g(Q_\infty^{\rho_1}, \rho_1) - g(Q_\infty^{\rho_2}, \rho_2) \geq -C_1(\rho_1 - \rho_2). \tag{B8}$$

First consider the case $\rho_1 > \rho_2$. Now, if the update in (13) is employed in the algorithm with $\rho^k \equiv \rho_*$, Theorem 3.5 in Abounadi et al. (2001) implies that Q^k will converge to some vector W. Consequently, $Q(i^*, a^*)$ will converge to some finite value $(W(i^*, a^*))$ and ρ^k will converge to the average reward of the policy defined in W. Then, from Proposition 2.10, one can conclude that $\rho^\infty = \rho_*$, which implies from (B5) that $g(Q_\infty^{\rho_*}, \rho_*) = 0$. Thus if, $\rho_2 = \rho_*$ and $\rho_1 = \rho^k$, inequality (B7) will lead to:

$$-C_2\Delta^k \leq g(Q_\infty^{\rho^k}, \rho^k) \leq -C_1\Delta^k.$$

Because $\beta(k) > 0$, the above leads to:

$$-C_2\Delta^k\beta(k) \leq \beta(k)g(Q_\infty^{\rho^k}, \rho^k) \leq -C_1\Delta^k\beta(k).$$

The above combined with (B6) leads to:

$$(1 - C_2\beta(k))\Delta^k + \beta(k)\delta^k \leq \Delta^{k+1} \leq (1 - C_1\beta(k))\Delta^k + \beta(k)\delta^k.$$

By iterating the above, one obtains for any $k > K$:

$$\prod_{i=K}^{i=k+1}(1 - C_2\beta(i))\Delta^K + \sum_{i=K}^{k}\prod_{j=i+1}^{k}(1 - C_2\beta(j))\beta(i)\delta^i \leq \Delta^{k+1}$$

$$\leq \prod_{i=K}^{i=k+1}(1 - C_1\beta(i))\Delta^K + \sum_{i=K}^{k}\prod_{j=i+1}^{k}(1 - C_1\beta(j))\beta(i)\delta^i. \qquad \text{(B9)}$$

Then, using (B2) and (B3), one has that as $k \to \infty$, $\Delta^{k+1} \to 0$, i.e., $\rho^k \to \rho_*$. Identical arguments can now be repeated for the case $\rho_2 > \rho_1$, i.e., for (B8), to obtain the same conclusion. □

One is now at a position to present the proof of Theorem 3.3, which will use Lemmas B.3 through B.4 presented above.

Proof: (Theorem 3.3) First note that $\rho^k \to \rho^\infty = \rho_*$ (from Lemmas B.3 and B.4). Now, if one replaces the term $r(i, a, j)$ of Theorem 3.5 in Abounadi et al. (2001) by $\left[r(i, a, j) - \theta(r(i, a, j) - \rho_*)^2\right]$, one has that $Q_{\rho_*}^k(i^*, a^*)$ will be well-defined, and therefore one can invoke that result. Then, Lemmas B.1 and B.2, together with the two time-scale convergence result and Theorem 3.5 of Abounadi et al. (2001), imply that Q^k will converge to a critical point of the ODE in (21), i.e., $Q^k \to Q_{\rho^\infty}^\infty$, where Q_ρ^∞ was defined in (22). But, $Q_{\rho^\infty}^\infty = Q_{\rho_*}^\infty$. Further, ρ_* equals the average reward of the policy defined in $Q_{\rho_*}^\infty$. Then, one can infer that the Q-factors converge will converge to a solution defined in Assumption 2.9 with $\phi_* = Q_{\rho_*}^\infty(i^*, a^*)$. Then, from Proposition 2.10, one has that $\{\mu^k\}_{k=1}^\infty$ converges to μ_*. □

Appendix C Proof of Theorem 4.3

Proof: The proof is very similar to the proof above for Theorem 3.3. But, the iterates must now be analyzed under asynchronous and noisy conditions and account for the difference in how ρ^k is updated. The proof will be presented as a sequence of Lemmas.

Define $f_{i,a}(.,.)$ of (19) as follows for the RL algorithm:

$$f_{i,a}(Q^k,\rho^k)(Q^k,\rho^k) = F_{i,a}(Q^k,\rho^k) - Q^k(i,a) + L_Q^k(i,a) \text{ where } L_Q^k(i,a) =$$

$$\left[r(i,a,\kappa_a^i) - \theta(r(i,a,\kappa_a^i) - \rho^k)^2 + \max_{b\in\mathcal{A}(\kappa_a^i)} Q^k(\kappa_a^i,b) - Q^k(i^*,a^*)\right] - F_{i,a}(Q^k,\rho^k)$$

in which κ_a^i denotes the random state reached when a is selected in i. This allows one to express the update of the Q-factors as: $Q^{k+1}(i,a) = Q^k(i,a) + \alpha(k)f_{i,a}(Q^k,\rho^k)$, which satisfies the format in (19). It is not hard to show that $\{L_Q^k(i,a)\}_{k=1}^{\infty}$ is a martingale sequence for a fixed value of ρ and every (i,a) pair.

For the RL algorithm, $\psi = r(i,a,\kappa_a^i)$. Define $g(.,.)$ of (20) as follows for the RL algorithm:

$$g(Q^k,\rho^k) = G(Q^k,\rho^k) - \rho^k + L_\rho^k \text{ where } L_\rho^k = r(i,a,\kappa_a^i) - G(Q^k,\rho^k) \text{ and}$$

$$G(Q^k,\rho^k) = \sum_{j\in\mathcal{S}} p(i,a,j)r(i,a,j) \text{ in which } a \in \arg\max_{b\in\mathcal{A}(i)} Q(i,b).$$

One can now write the update of the slower iterate, ρ^k, as:

$$\rho^{k+1} = \rho^k + \beta(k)g(Q^k,\rho^k), \tag{C1}$$

which satisfies the format in (20). It is easy to show that $\{L_\rho^k\}_{k=1}^{\infty}$ is also a martingale sequence. It is standard in the literature on RL to show that the effect of the martingale terms vanishes in the limit, provided the sequence $\{Q^k,\rho^k\}_{k=1}^{\infty}$ remains bounded. Hence, the boundedness of this sequence via the following Lemma will be first shown.

Lemma C.1: *$\{\rho^k\}_{k=1}^{\infty}$ remains bounded, and $\{Q^k\}_{k=1}^{\infty}$ remains bounded almost surely.*

Proof: The proof is similar to Lemma B.1 noting that $\{Q^k\}_{k=1}^{\infty}$ will remain bounded almost surely in this case. □

Since the noise term vanishes in the limit, the $f_{i,a}(,.,)$ term of the RL algorithm can be analyzed similarly to that in the DP algorithm. Further, the exploratory strategy can be shown to ensure that all state-action pairs are visited infinitely often in the limit (as discussed above). Then, if the Q-factors are updated using $\rho^k \equiv \rho_*$, one should have that $\rho^k \to \rho_*$, since ρ^k estimates the average reward of the policy contained in the Q-factors. Additionally, Lemma B.3 and B.2 (already proved) apply for the RL algorithm.

Lemma C.2: *$\rho^k \to \rho_*$ almost surely when Assumptions 2.9, 3.1, and 3.2 hold and $\alpha(k)$ and $\beta(k)$ satisfy Assumptions 4.1 and 4.2.*

Proof: The proof will be similar to that of Lemma B.4 for the DP algorithm with some differences. First, one needs to set $\rho_1 = \rho_*$. Second, the updating will be asynchronous here. Fortunately, asynchronously, the Q-factors will track a *scaled* version of the ODE tracked by their synchronous counterparts, and furthermore, the scaled ODE and the DP algorithm's ODE will have the same critical point and trajectory (Borkar, 2008, Chap 7.). Then, the analysis in the proof of Lemma B.4 can be mimicked to obtain the result. □

Convergence to the optimal policy follows similarly as in the proof of Theorem 3.3. □