Proceedings of the 2009 Winter Simulation Conference
M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin, and R. G. Ingalls, eds.

REINFORCEMENT LEARNING FOR MODEL BUILDING AND VARIANCE-PENALIZED CONTROL

Abhijit Gosavi

Department of Engineering Management and Systems Engineering
219 Engineering Management
Missouri University of Science and Technology
Rolla, MO 65409, USA

ABSTRACT

Reinforcement learning (RL) is a simulation-based technique to solve Markov decision problems or processes (MDPs). It is especially useful if the transition probabilities in the MDP are hard to find or if the number of states in the problem is too large. In this paper, we present a new model-based RL algorithm that builds the transition probability model without the generation of the transition probabilities; the literature on model-based RL attempts to compute the transition probabilities. We also present a variance-penalized Bellman equation and an RL algorithm that uses it to solve a variance-penalized MDP. We conclude with some numerical experiments with these algorithms.

1 INTRODUCTION

For the Markov decision problem (MDP), Dynamic Programming (DP) can be used for solution purposes. The method of Reinforcement Learning (RL) or approximate DP (ADP) has emerged from some seminal works that use value iteration (Watkins 1989) or policy iteration (Witten 1977; Barto, Sutton, and Anderson 1983; Werbös 1987). RL (Bertsekas and Tsitsiklis 1996, Sutton and Barto 1998, Gosavi 2003) is a simulation-based method that uses the Robbins-Monro algorithm. It has attracted a great deal of interest recently because it generates near-optimal solutions to the MDP without the apriori knowledge of its transition probability model. In this paper, (i) we develop a new type of model-building RL algorithm and (ii) explore the use of RL for solving a variance-penalized MDP. Both algorithms use a two-time-scale framework developed in Borkar (1997).

For model-based RL, we develop algorithms for discounted and average reward MDPs that indirectly build the transition probability model without counting the number of visits to states or using the Bayesian networks — typically needed for model-based RL (Tadepalli and Ok 1998). Our algorithms have two advantages: (i) They can be combined with artificial neural networks (ANNs) (Werbös 1974), and (ii) they avoid the computation of an expectation in every iteration, which is required in model-based RL and is computationally intensive for large state-spaces. Regarding function approximation, we need to point out that effective function approximation lies at the heart of a successful application of RL, and this has been recognized since the early days of RL (Werbös 1987). ANNs have been criticized in the literature as being unstable for function approximation with RL (Baird 1995, Sutton 1996, Tsitsiklis and Roy 1997). Unfortunately, much of the literature does not investigate the real causes for the instability of the ANN, although it is quite likely that there are many. One of the reasons for the failure of ANNs in RL is the noise inherent in most RL algorithms like Q-Learning (Watkins 1989) or SARSA (Rummery and Niranjan 1994) that are "model-free." Interestingly, this has been highlighted some time back in some insightful work by Werbös (1990) and Williams (1988). Our model-based algorithm seeks to eliminate from the picture the noise of model-free algorithm that destabilizes the ANN.

For risk-sensitive control, we employ variance as a measure of risk, originally conceived of in Markowitz (1952), within the MDP framework. We study variance because it is heavily used in the world of finance and is well-understood by real-world managers seeking risk-sensitive solutions for strategic planning. Our model-free RL algorithm is developed for a problem solvable via a quadratic program (see the seminal work of Filar, Kallenberg, and Lee (1989)), when the transition probability model is known. We also present a semi-Markov extension of our algorithm. The problem we study has been studied via

a policy gradient approach in Sato and Kobayashi (2001). Other approaches that use different metrics for modeling risk in RL include Borkar (2002) and Geibel and Wysotzki (2005).

We have organized the remainder of this paper as follows. In Section 2, we describe the new algorithms for model building, and in Section 3, we present algorithms for variance-penalized control. In Section 4, we present numerical examples for our algorithms. Section 5 concludes the paper.

2 MODEL-BUILDING RL

We first present some notation and then present our model-building algorithms.

2.1 Notation

The goal in an MDP is to optimize a function of the immediate reward. Let $\mathcal{S}$ denote the finite set of states in the MDP, $\mathcal{A}(i)$ the finite set of actions permitted in state i, and $d(i)$ the action chosen in state i when policy $\hat{d}$ is pursued, where $\cup_{i\in\mathcal{S}}\mathcal{A}(i)=\mathcal{A}$. Further let $r(.,.,.):\mathcal{S}\times\mathcal{A}\times\mathcal{S}\rightarrow\Re$ denote the one-step immediate reward and $p(.,.,.):\mathcal{S}\times\mathcal{A}\times\mathcal{S}\rightarrow[0,1]$ denote the associated transition probability. Then the *expected* immediate reward earned in state i when action a is chosen in it can be defined as: $\bar{r}(i,a)=\sum_{j=1}^{|\mathcal{S}|}p(i,a,j)r(i,a,j)$.

Definition 1 *The long-run average (expected) reward of a policy $\hat{d}$ starting at state i in an MDP is: $\rho_{\hat{d}}(i)\equiv\lim_{k\rightarrow\infty}E_{\hat{d}}\left[\sum_{s=1}^{k}\bar{r}(z_s,d(z_s))\middle|z_1=i\right]/k$, where z_s is the state occupied before the sth transition and $E_{\hat{d}}$ denotes the expectation induced by $\hat{d}$.*

Definition 2 *The long-run discounted reward of a policy $\hat{d}$ starting at state i in an MDP is: $\kappa_{\hat{d}}(i)\equiv\lim_{k\rightarrow\infty}E_{\hat{d}}\left[\sum_{s=1}^{k}\gamma^{s-1}\bar{r}(z_s,d(z_s))\middle|z_1=i\right]/k$, where γ is the discount factor.*

We note that the Bellman equations (in terms of Q-values), which our algorithms seek to solve, are as follows. For the average reward MDP, it is:

$$Q(i,a)=\sum_{j\in\mathcal{S}}p(i,a,j)\left[r(i,a,j)+\max_{b\in\mathcal{A}(j)}Q(j,b)-\rho^*\right]\quad\forall(i,a),\tag{1}$$

where ρ^* is the optimal average reward, and for the discounted MDP, it is:

$$Q(i,a)=\sum_{j\in\mathcal{S}}p(i,a,j)\left[r(i,a,j)+\gamma\max_{b\in\mathcal{A}(j)}Q(j,b)\right]\quad\forall(i,a).\tag{2}$$

2.2 Model-Building Algorithms

As stated above, model-building may hold the key to solving the problem of instability of the ANN in combination with RL (Werbös 1990, Williams 1988). It is also known that model-based algorithms are oftentimes more stable and require less experimenting with learning rates to find a suitable one. In the literature, model-building RL either depends on straightforward counting (Tadepalli and Ok 1998) or exploits a Bayesian network. We present an algorithm that does not need a counting mechanism or a Bayesian net, and can be combined with function approximators like ANNs. We present an algorithm for discounted reward MDPs, with γ as the discounting factor, (Alg. A-1), and then an algorithm for average reward MDPs (Alg. A-2).

Steps in Algorithm A-1:

Step 1. Set for all (l,u), where $l\in\mathcal{S}$ and $u\in\mathcal{A}(l)$, $Q(l,u)\leftarrow 0$, $\tilde{r}(l,u)\leftarrow 0$, and $\tilde{Q}(l,u)\leftarrow 0$. Note that $\tilde{r}(l,u)$ denotes the estimate of the expected immediate reward in state l when action u is chosen, and $\tilde{Q}(l,u)$ denote the estimate of the maximum Q-value for the next state when action u is chosen in state l. Set k, the number of state changes, to 0. Set $k_{\max}$, which denotes the maximum number of iterations for which the algorithm is run, to a sufficiently large number; note that the algorithm runs iteratively between Steps 2 and 6. Start system simulation at any arbitrary state.

Step 2. Let the current state be i. Select action a with a probability of $1/|\mathcal{A}(i)|$ or with some other rule such as the Boltzmann selection rule.

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. Increment k by 1.

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

$$Q^{k+1}(i,a) \leftarrow (1-\mu)Q^k(i,a) + \mu\left[\tilde{r}(i,a) + \gamma\tilde{Q}^k(i,a)\right].$$

Step 5. Update $\tilde{Q}(i,a)$ and $\tilde{r}(i,a)$ as follows:

$$\tilde{Q}^{k+1}(i,a) \leftarrow (1-\nu)\tilde{Q}^k(i,a) + \nu \max_{b\in\mathcal{A}(j)} Q^k(j,b); \quad \tilde{r}(i,a) \leftarrow (1-\nu)\tilde{r}(i,a) + \nu r(i,a,j).$$

Step 6. If $k < k_{max}$, set $i \leftarrow j$ and then go to Step 2. Otherwise, go to Step 7.

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

Note that the updates in Steps 4 and 5 do not use samples, but instead use estimates of both the immediate reward function and the Q-value. As such, our algorithm is model-based; but unlike classical model-based algorithms it does not estimate the transition probabilities directly. It thus avoids the noise of the model-free algorithm and at the same time can be handily combined with ANNs without the need for Bayesian nets.

Steps in Algorithm A-2:

Algorithm A-2 will handle the average reward case; the steps are same as in A-1 with the following changes: Any one state-action pair, to be denoted by (i^*,a^*), is selected in Step 1, and the update in Step 4 is changed to the following:

$$Q^{k+1}(i,a) \leftarrow (1-\mu)Q^k(i,a) + \mu\left[\tilde{r}(i,a) + \tilde{Q}^k(i,a) - \tilde{Q}(i^*,a^*)\right].$$

3 VARIANCE-PENALIZED RL

We first present some relevant notation. Thereafter, we present two algorithms, B-1 and B-2.

3.1 Notation

The goal in a variance-penalized problem is to optimize a mean-variance function in which the variance of revenues is penalized.

Definition 3 *The long-run variance of the reward of a policy $\hat{d}$ starting at state i in an MDP is:* $\psi_{\hat{d}}(i) \equiv \lim_{k\to\infty} E_{\hat{d}}\left[\sum_{s=1}^{k}\left[\bar{r}(z_s,d(z_s)) - \rho_{\hat{d}}\right]^2 \middle| z_1 = i\right] \Big/ k$ *(Filar, Kallenberg, and Lee 1989).*

It can be shown that both $\rho_{\hat{d}}(.)$ and $\psi_{\hat{d}}(.)$ are independent of the starting state for irreducible and recurrent Markov chains. In our notation, $\vec{x}$ will denote a column vector whose ith element is $x(i)$. A policy that maximizes

$$\phi(i) \equiv \rho_{\hat{d}}(i) - \theta\psi_{\hat{d}}(i)$$

for every $i \in \mathcal{S}$ is optimal in the variance-penalized framework (Filar, Kallenberg, and Lee 1989). The *variance-penalized score* $\phi(i)$ can also be shown to be independent of i.

Let us consider the following equation from Gosavi (2007). For all $(i,a) \in \mathcal{S} \times \mathcal{A}(i)$,

$$Q(i,a) = \sum_{j\in\mathcal{S}} p(i,a,j)\left[r(i,a,j) - \theta(r(i,a,j)-\rho)^2 - \phi + \max_{b\in\mathcal{A}(j)} Q(j,b)\right]. \tag{3}$$

We will assume that a solution to this equation exists, and that the Q-values that solve this equation define a policy whose average reward is ρ and variance-penalized score is ϕ. We will refer to the above (i.e., (3)) as the *variance-penalized Bellman equation* (Gosavi 2007). Whether a solution exists is still being investigated (Gosavi and Meyn 2009), but in practice solving this equation via our algorithm invariably leads one to the optimal solution. That a deterministic optimal solution exists to this problem has been proved in Filar, Kallenberg, and Lee (1989) via a quadratic program, and the optimal solution to the problem can be determined via exhaustive enumeration of all deterministic policies or by solving the quadratic program.

For the semi-Markov decision problem (SMDP), we need some more notation. Let $t(i,a,j)$ denote the time taken for a transition from i to j when action a is selected in i. Also, let $\bar{t}(i,a) = \sum_{j=1}^{|S|} p(i,a,j)t(i,a,j)$. Then we define three quantities:

$$\alpha_{\hat{d}}(i) \equiv \lim_{k\to\infty} E_{\hat{d}} \left[\sum_{s=1}^{k} \bar{r}(z_s, d(z_s)) | z_1 = i \right] \Big/ k,$$

which is the first moment of the immediate reward,

$$\beta_{\hat{d}}(i) \equiv \lim_{k\to\infty} E_{\hat{d}} \left[\sum_{s=1}^{k} \bar{t}(z_s, d(z_s)) | z_1 = i \right] \Big/ k,$$

which is the first moment of the time in one transition, and

$$\gamma_{\hat{d}}(i) \equiv \lim_{k\to\infty} E_{\hat{d}} \left[\sum_{s=1}^{k} \bar{r}^2(z_s, d(z_s)) | z_1 = i \right] \Big/ k,$$

which is the second moment of the immediate reward. Then for irreducible and recurrent Markov chains, the long-run average reward of a policy $\hat{d}$ in an SMDP, starting at state i, is

$$\rho_{\hat{d}}(i) = \frac{\alpha_{\hat{d}}(i)}{\beta_{\hat{d}}(i)},$$

and via Theorem 1 of Gosavi (2006), the long-run variance of rewards of the policy $\hat{d}$ in an SMDP, starting at state i, can be defined as:

$$\psi_{\hat{d}}(i) = \frac{\gamma_{\hat{d}(i)}}{\beta_{\hat{d}}(i)} - \frac{(\alpha_{\hat{d}}(i))^2}{\beta_{\hat{d}}(i)}.$$

The Bellman equation for the SMDP is proposed as follows: For all $(i,a) \in \mathscr{S} \times \mathscr{A}(i)$,

$$Q(i,a) = \sum_{j\in\mathscr{S}} p(i,a,j) \left[r(i,a,j) - \theta(r(i,a,j) - \rho)^2 - \phi t(i,a,j) + \max_{b\in\mathscr{A}(j)} Q(j,b) \right],$$

where ρ denotes the optimal average reward on a *per transition* basis (not on unit time basis) and ϕ denotes the optimal score on a unit time basis.

3.2 Algorithms

Algorithm B-1 is for the MDP and B-2 for the SMDP. We now present the details.

Steps in Algorithm B-1:

Step 1. Set k, the number of state changes, to 0. Set for all (l,u), where $l \in \mathscr{S}$ and $u \in \mathscr{A}(l)$, $Q^k(l,u) \leftarrow 0$. Set ρ^k, the estimate of the long-run reward per state change, and σ^k, the estimate of the long-run squared reward per state change, to 0. Set $k_{\max}$, which is the number of iterations for which the algorithm is run, to a sufficiently large number;note that the algorithm is run iteratively between Steps 2 and 6. Start system simulation at any arbitrary state.

Step 2. Let the current state be i. Select action a with a probability of $1/|\mathscr{A}(i)|$ or with some other rule such as the Boltzmann selection rule. Action a will be considered greedy if $a = \arg\max_{b\in\mathscr{A}(i)} Q^k(i,b)$.

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. Increment k by 1. Then calculate μ and ν, both of which are predefined functions of k.(See Remark below).

Step 4. Compute $\psi^k = \sigma^k - (\rho^k)^2$ and $\phi^k = \rho^k - \theta\psi^k$. Update $Q(i,a)$ as follows:

$$Q^{k+1}(i,a) \leftarrow (1-\mu(k))Q^k(i,a) + \mu(k) \left[r(i,a,j) - \theta \left(r(i,a,j) - \rho^k \right)^2 - \phi^k + \max_{b\in\mathscr{A}(j)} Q^k(j,b) \right].$$

Step 5. If a is greedy, update ρ and σ using the following:

$$\rho^{k+1} \leftarrow (1-\nu(k))\rho^k + \nu(k)\frac{[r(i,a,j)+\rho^k k]}{k+1}; \sigma^{k+1} \leftarrow (1-\nu(k))\sigma^k + \nu(k)\frac{[(r(i,a,j))^2+\sigma^k k]}{k+1}.$$

Step 6. If $k < k_{max}$, set $i \leftarrow j$ and then go to Step 2. Otherwise, go to Step 7.
Step 7. For each $l \in \mathscr{S}$, select $d(l) \in \arg\max_{b\in\mathscr{A}(l)} Q^k(l,b)$. The policy returned is $\hat{d}$. Stop.
Remark: Note that $\limsup_{k\to\infty} \nu(k)/\mu(k) = 0$ is a condition required of the step-sizes (Borkar 1997).

The three main differences with the steps of B-1 for B-2 (the algorithm for the SMDP) will be:
Step 1: In addition to other tasks, let τ^k denote the estimate of the time spent in each transition, which should be initialized to a very small positive quantity.
Step 4: Update ψ^k as in B-1, but ϕ^k as follows:

$$\phi^k = [\rho^k - \theta\psi^k]/\tau^k.$$

Update $Q(i,a)$ as follows:

$$Q^{k+1}(i,a) \leftarrow (1-\mu(k))Q^k(i,a) + \mu(k)\left[r(i,a,j) - \theta\left(r(i,a,j)-\rho^k\right)^2 - \phi^k t(i,a,j) + \max_{b\in\mathscr{A}(j)} Q^k(j,b)\right].$$

Step 5: In addition to updating ρ and σ, update τ^k in case of a greedy update as follows:

$$\tau^{k+1} \leftarrow (1-\nu(k))\tau^k + \nu(k)\left[t(i,a,j)+\tau^k k\right]\Big/(k+1).$$

4 NUMERICAL EXPERIMENTS

In this section, we present a sample of our numerical experiments to illustrate the usefulness of our algorithms. We first present the results with the model-building algorithm and then present the same for the variance-penalized algorithm.

We tested our algorithms on a 2-state MDP in which two actions allowed per state in which all Markov chains are regular. $\mathbf{P_a}$ and $\mathbf{R_a}$ denote the transition probability and reward matrices for action a respectively; $\mathbf{P_a}(i,j) = p(i,a,j)$ and $\mathbf{R_a}(i,j) = r(i,a,j)$.

$$\mathbf{P}_1 = \begin{bmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{bmatrix}; \mathbf{P}_2 = \begin{bmatrix} 0.9 & 0.1 \\ 0.2 & 0.8 \end{bmatrix}; \mathbf{R}_1 = \begin{bmatrix} 6.0 & -5 \\ 7.0 & 12 \end{bmatrix}; \mathbf{R}_2 = \begin{bmatrix} 5.0 & 68 \\ -2 & 12 \end{bmatrix}. \quad (4)$$

Table 1: For A-2, $\tilde{Q}(i^*,a^*) \equiv \tilde{Q}(1,1) = 10.55689 \approx 10.56 =$ optimal average reward

	$\gamma = 0.8; \theta = 0$		$\theta = 0$
$Q(.,.)$	A-1	Q-Value-iteration	A-2
Q(1,1)	42.668598	44.84667	2.70674
Q(1,2)	50.959958	53.03333	10.98493
Q(2,1)	49.230419	51.86667	9.581180
Q(2,2)	46.284741	49.28000	6.672185

Example I: This is used to test algorithms A-1 and A-2. We changed $\mathbf{R_2}$ in (4) as follows: $r(1,2,1) = 10$; $r(1,2,2) = 17$; $r(2,2,1) = -14$; $r(2,2,2) = 13$.
Example II: This example is designed to test algorithms B-1 and B-2. The data in (4) is used.

Table 1 shows the Q-values obtained from the algorithms, A-1, A-2, and a discounted value iteration algorithm based on Q-values (Example I). Each algorithm is run for a maximum of 100 iterations in a simulator; the Q-values remained bounded in all our experiments. The step-size rules used were: $\mu^k = \frac{\log(k+1)}{k+1}$; $\eta^k = \frac{10}{k+9}$. Table 2 lists the mean and variance via exhaustive enumeration for Examples I and II.

Table 2: The optimal policy's metrics are in bold

	Example II ($\theta = 0.15$)			Example I ($\theta = 0$)
Policy	ρ	ψ	ϕ	ρ
(1,1)	5.828571	30.142041	1.307265	5.828571
$(1,2)$	8.625000	31.284375	**3.932344**	5.64
(2,1)	11.04000	287.23840	-32.04576	**10.56**
(2,2)	10.95000	187.54750	-17.182125	9.666667

5 CONCLUSIONS

In this paper, we develop a model-free RL algorithm for solving the variance-penalized MDP and a model-building RL algorithm that does not need explicit counting or Bayesian nets but (a) is compatible with ANNs and (b) does not compute an expectation in every iteration. Although we presented only a sample here, our numerical results are encouraging since they solve the classical Bellman equation (which is (1) or (2)) and the variance-penalized Bellman equation (which is (3)). What remains as a topic for future work is a study of the convergence properties of these algorithms. We believe that the model-building algorithm has the potential to overcome some problems associated with combining RL with ANNs. Experiments need to be performed to test whether this is true of large-scale problems.

ACKNOWLEDGMENTS

This research was supported in part by the grant ECS:0841055 awarded by the National Science Foundation.

REFERENCES

Baird, L. C. 1995. Residual algorithms: Reinforcement learning with function approximation. In *Proceedings of the Twelfth International Conference on Machine Learning*.

Barto, A., R. Sutton, and C. Anderson. 1983. Neuronlike elements that can solve difficult learning control problems. *IEEE Transactions on SMC* 13:835–846.

Bertsekas, D., and J. Tsitsiklis. 1996. *Neuro-dynamic programming*. Belmont, MA, USA: Athena Scientific.

Borkar, V. 2002. Q-learning for risk-sensitive control. *Mathematics of Operations Research* 27(2):294–311.

Borkar, V. S. 1997. Stochastic approximation with two-time scales. *Systems and Control Letters* 29:291–294.

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

Geibel, P., and F. Wysotzki. 2005. Risk-sensitive reinforcement learning applied to control under constraints. *Journal of Artificial Intelligence Research* 24:81–108.

Gosavi, A. 2003. *Simulation-based optimization: Parametric optimization techniques and reinforcement learning*. Boston, MA: Kluwer Academic.

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, Dallas, TX*.

Gosavi, A., and S. Meyn. 2009. A dynamic programming algorithm for variance-penalized Markov decision process. Working Paper, Missouri University of Science and Technology and University of Illinois.

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

Rummery, G., and M. Niranjan. 1994. On-line Q-learning using connectionist systems. Technical Report CUED/F-INFENG/TR 166. Engineering Department, Cambridge University.

Sato, M., and S. Kobayashi. 2001. Average-reward reinforcement learning for variance-penalized Markov decision problems. In *Proceedings of the 18th International Conference on Machine Learning*, 473–480. Morgan Kauffman.

Sutton, R. 1996. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in Neural Information Processing Systems 8*. Cambridge, MA: MIT Press.

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

Tadepalli, P., and D. Ok. 1998. Model-based Average Reward Reinforcement Learning Algorithms. *Artificial Intelligence* 100:177–224.

Tsitsiklis, J. N., and B. V. Roy. 1997. An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control* 42(5):674–690.

Watkins, C. 1989. *Learning from delayed rewards*. Ph. D. thesis, Kings College, England.

Werbös, P. 1990. A menu of designs for reinforcement learning over time. In *Neural Networks for Control*, 67–95. MIT Press, MA.

Werbös, P. J. 1974, May. *Beyond regression: New tools for prediction and analysis of behavioral sciences*. Ph. D. thesis, Harvard University, Cambridege, MA, USA.

Werbös, P. J. 1987. Building and understanding adaptive systems: A statistical/numerical approach to factory automation and brain research. *IEEE Transactions on SMC* 17:7–20.

Williams, R. 1988. On the use of backpropagation in associative reinforcement learning. In *Proceedings of the International Conference on Neural Networks, San Diego, CA*.

Witten, I. 1977. An adaptive optimal controller for discrete time Markov environments. *Information and Control* 34:286–295.

AUTHOR BIOGRAPHY

ABHIJIT GOSAVI is an Assistant Professor in the Engineering Management and Systems Engineering Department at the Missouri University of Science and Technology. He has published numerous papers in journals such as *Automatica*, *INFORMS Journal on Computing*, *Machine Learning*, and *Systems and Control Letters*. He is also the author of the textbook, *Simulation-Based Optimization: Parametric Optimization Techniques and Reinforcement Learning*, published by Kluwer (now Springer). His email address for these proceedings is <gosavia@mst.edu>.