Proceedings of the 2013 Winter Simulation Conference
R. Pasupathy, S.-H. Kim, A. Tolk, R. Hill, and M. E. Kuhl, eds.

RELATIVE VALUE ITERATION FOR AVERAGE REWARD SEMI-MARKOV CONTROL VIA SIMULATION

Abhijit Gosavi

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

ABSTRACT

This paper studies the semi-Markov decision process (SMDP) under the long-run average reward criterion in the simulation-based context. Using dynamic programming, a straightforward approach for solving this problem involves policy iteration; a value iteration approach for this problem involves a transformation that induces an additional computational burden. In the simulation-based context, however, where one seeks to avoid the transition probabilities needed in dynamic programming, value iteration forms a more convenient route for solution purposes. In this paper, hence, we present (to the best of knowledge for the first time) a relative value iteration algorithm for solving average reward SMDPs via simulation. The algorithm is a semi-Markov extension of an algorithm in the literature for the Markov decision process. Our numerical results with the new algorithm are very encouraging.

1 INTRODUCTION

Simulation-based methods have now become quite popular for solving Markov decision processes/problems (MDPs). In this paper, our interest lies in the semi-Markov decision process (SMDP), which is a more generalized version of the MDP. In the SMDP, the time spent in each transition of the underlying Markov chain is an integral part of the model and indeed of the objective function. Our focus in this work is on the average reward criterion for an infinite horizon SMDP. Further, our interest is in simulation-based methods that can avoid the transition probabilities underlying these problems that can be very difficult to determine for many real-world systems. The theory of dynamic programming (DP) that seeks to solve MDPs/SMDPs when the transition probabilities are available has been discussed in a number of books, including Bertsekas (2012) and Puterman (1994).

Simulation-based methods for solving MDPs/SMDPs also go by the name reinforcement learning (Bertsekas and Tsitsiklis 1996, Sutton and Barto 1998), often abbreviated as RL. Recently, there has been a great deal of interest in simulation-based optimization in general. Textbooks that study *simulation-based optimization* for MDPs/SMDPs include Gosavi (2003) and Chang et al. (2007). There are numerous other books that cover this topic, and the reader is referred to Bertsekas (2012) for a comprehensive account of the topic and numerous references.

The central idea in RL is to use a simulator of the system and employ a step-size-based version of a DP algorithm (the latter uses expectations over the transition probabilities). These step-size-based algorithms require samples in its updates rather than expectations, and these samples can be handily gathered within simulators. Note, however, that in general, simulation-based optimization for these problems can use other techniques that are *not* based on DP; Chang et al. (2007) discusses a number of such techniques amongst others.

The SMDP is a generalized version of the MDP in which the time spent in each transition of the underlying Markov chains is allowed to be a random variable. In the MDP, the time of transition is not relevant to the objective function and is hence assumed to be unity. However, in the SMDP, the transition time of each transition is not necessarily the same, and as stated above, is an integral part of the objective function. For the average reward SMDP, RL algorithms have been proposed in the literature (Das et al. 1999; Gosavi 2004); however, these algorithms require a separate update of the average reward. An algorithm for MDPs has been proposed in Abounadi et al. (2001) that performs a *relative* value iteration within the simulator, and the algorithm avoids having to update the average reward separately. Hence, in this paper, we study a relative value iteration algorithm for SMDPs that does not need a separate update of the average reward. We obtained encouraging numerical results with the DP and RL versions of this algorithm on small-scale problems. Future work will involve application on large-scale problems and a convergence analysis.

The rest of this paper is organized as follows. In Section 2, we present a mathematical background to SMDPs, some notation, and the associated Bellman equation. In Section 3, we present the DP and the RL algorithms. Numerical results with the algorithms are provided in Section 4. Section 5 concludes the paper with a discussion of the scope for future work.

2 BACKGROUND

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
- μ: A deterministic policy
- $\mu(i)$: The action chosen in state i under policy μ
- $p(i,a,j)$: The probability of one transition from i to j when action $a \in \mathscr{A}(i)$ is selected
- $r(i,a,j)$: The (non-random) immediate reward earned in one transition from i to j when action $a \in \mathscr{A}(i)$ is selected
- $t(i,a,j)$: The time (possibly random) spent in one transition from i to j when action $a \in \mathscr{A}(i)$ is selected
- $\bar{r}(i,a) = \sum_{j \in \mathscr{S}} p(i,a,j)r(i,a,j)$: The mean reward earned in one transition from i to j when action $a \in \mathscr{A}(i)$ is selected
- $\bar{t}(i,a,j)$: The mean time spent in one transition from i to j when action $a \in \mathscr{A}(i)$ is selected
- $\bar{t}(i,a) = \sum_{j \in \mathscr{S}} p(i,a,j)\bar{t}(i,a,j)$: The mean time needed for one transition from i when action $a \in \mathscr{A}(i)$ is selected
- $\mathbf{P_a}$: Transition probability matrix (TPM) when action a is selected in every state; the (i,j)th element of this matrix is denoted by $P_a(i,j)$ and equals $p(i,a,j)$
- $\mathbf{R_a}$: Transition reward matrix (TRM) when action a is selected in every state; the (i,j)th element of this matrix is denoted by $R_a(i,j)$ and equals $r(i,a,j)$
- $\mathbf{T_a}$: Transition time matrix (TTM) when action a is selected in every state; the (i,j)th element of this matrix is denoted by $T_a(i,j)$, where the latter denotes the *mean* time spent in one transition from i to j when action a is selected and equals $\bar{t}(i,a,j)$

The average reward for an infinite horizon SMDP (Bertsekas 2012), starting from state i, of the policy μ can be mathematically expressed as:

$$\rho_\mu(i) = \liminf_{K\to\infty} \frac{\mathsf{E}[\sum_{k=1}^{K} r(x_k,\mu(x_k),x_{k+1})|x_1 = i]}{\mathsf{E}[\sum_{k=1}^{K} t(x_k,\pi(x_k),x_{k+1})|x_1 = i]},$$

where x_k is the state from which the kth jump of the Markov chain occurs in the trajectory and $\mathsf{E}[.]$ denotes the expectation over the trajectory. It is assumed here that the Markov chain associated with any policy μ in the problem is regular (Grinstead and Snell 1997). Then the limiting (or steady-state or invariant) probabilities of the Markov chain exist. Under this condition, the average reward of the policy is independent of the starting state i, and hence $\rho_\mu(i)$ for any $i \in \mathscr{S}$ can be replaced by ρ_μ.

The following well-known result (see e.g., Vol II of Bertsekas (2012) for proof) establishes the Bellman optimality equation for the average reward SMDP:

Theorem 1 There exists a scalar ρ^* and a value function $v^* : \mathscr{S} \to \Re$ satisfying the following system of equations for all $i \in S$,

$$v^*(i) = \max_{a \in \mathscr{A}(i)} \left[\bar{r}(i,a) - \rho^* \bar{t}(i,a) + \sum_{j \in S} p(i,a,j) v^*(j) \right], \tag{1}$$

such that the *greedy* policy μ^* formed by selecting actions that maximize the right-hand side of the above equation is *average-reward optimal*.

Equation (1) is the Bellman optimality equation for average reward SMDPs. We will now present a Q-factor version of this equation. We first define the Q-factor in this context as follows. The Q-function is the function $Q : \mathscr{S} \times \mathscr{A} \to \Re$ where $\mathscr{A} = \cup_{i \in \mathscr{S}} \mathscr{A}(i)$ and is defined as:

$$Q(i,a) = \sum_{j \in \mathscr{S}} p(i,a,j) \left[r(i,a,j) - \rho^* \bar{t}(i,a,j) + v^*(j) \right] \quad \forall (i,a). \tag{2}$$

Note that we can write the Bellman equation above as:

$$v^*(i) = \max_{a \in \mathscr{A}(i)} \left[\sum_{j \in \mathscr{S}} p(i,a,j) \left[r(i,a,j) - \rho^* \bar{t}(i,a,j) + v^*(j) \right] \right] \quad \forall i. \tag{3}$$

From Equations (2) and (3), we have that:

$$v^*(i) = \max_{a \in \mathscr{A}(i)} Q(i,a) \quad \forall i.$$

The above allows us to express the Q-factor, $Q(i,a)$, as:

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

This allows us to rewrite Theorem 1 as:

Theorem 2 There exists a scalar ρ^* and a Q-function $Q : \mathscr{S} \times \mathscr{A} \to \Re$ satisfying the following system of equations for all $i \in S$ and all $a \in \mathscr{A}(i)$:

$$Q(i,a) = \sum_{j \in \mathscr{S}} p(i,a,j) \left[r(i,a,j) - \rho^* \bar{t}(i,a,j) + \max_{b \in \mathscr{A}(j)} Q(j,b) \right], \tag{4}$$

such that the policy μ^*, defined by $\mu^*(i) = \arg\max_{a \in \mathscr{A}(i)} Q(i,a)$ for every $i \in \mathscr{S}$, is average-reward optimal.

The above result will be the basis for developing DP and RL algorithms for the problem at hand.

3 DP AND RL ALGORITHMS

A value iteration algorithm based *directly* on the Bellman equation in Equation (1) is not feasible since ρ^* is unknown at the start. Hence, a "discretization" technique has been proposed in the literature that seeks to transform the SMDP into an MDP. In other words, a new Bellman equation is obtained that can then be used within a relative value iteration algorithm. This discretization approach works as follows: For all $i \in \mathscr{S}$,

$$v^*(i) \leftarrow \max_{a \in \mathscr{A}(i)} \left[\hat{r}(i,a) + \sum_{j \in S} \hat{p}(i,a,j) v^*(j) \right],$$

where

$$\hat{r}(i,a) = \bar{r}(i,a)/\bar{t}(i,a),$$

$$\hat{p}(i,a,j) = \chi p(i,a,j)/\bar{t}(i,a), \text{ if } i \neq j,$$

and

$$\hat{p}(i,a,j) = 1 + \chi[p(i,a,j) - 1]/\bar{t}(i,a), \text{ if } i = j.$$

In the above, χ should satisfy

$$0 \leq \chi \leq \bar{t}(i,a)/\{1 - p(i,a,i)\}$$

for all a, i and j. A relative value iteration algorithm can then be employed using the discretized Bellman equation. Although the algorithm based on the above has been alluded to in the literature (Bertsekas 2012, Puterman 1994), the specific steps have never been described. Since, our new algorithm seeks to be a competitor to it, we provide a detailed description of the steps.

Step 1. Set $k = 1$ and $v^k(i) = w^k(i) = 0$ for all $i \in \mathscr{S}$. Select any state in the system to be the distinguished state, i^*. Set ε to a small positive value.

Step 2. For all $i \in \mathscr{S}$, compute:

$$w^{k+1}(i) = \max_{a \in \mathscr{A}(i)} \left[\hat{r}(i,a) + \sum_{j \in S} \hat{p}(i,a,j) v^k(j) \right].$$

Step 3. For all $i \in \mathscr{S}$, compute:

$$v^{k+1}(i) = w^{k+1}(i) - w^{k+1}(i^*).$$

Step 4. Let v^k denote the vector of values in the kth iteration and v^{k+1} denote the same in the $(k+1)$th iteration. Check for the termination condition: if $||v^{k+1} - v^k||_\infty \geq \varepsilon$, increase k by 1 and return to Step 2. Otherwise, compute the ε-optimal policy, μ_ε, as follows: $\mu_\varepsilon(i) = \arg\max_{a \in \mathscr{A}(i)} \left[\hat{r}(i,a) + \sum_{j \in S} \hat{p}(i,a,j) v^k(j) \right]$ for every $i \in \mathscr{S}$, and stop.

One difficulty with the above procedure is that to compute the functions $\hat{p}(.,.,.)$ and $\hat{r}(.,.)$, one must experiment with various values of χ in order to determine one that meets the condition above. Further, the above approach requires change in the transition probabilities that we altogether seek to avoid in RL; hence the above algorithm does not directly suggest an RL algorithm — in which the simulator runs according to the original transition probabilities.

We now consider a more direct approach inspired by Q-Learning (Watkins 1989). We will use a step size version of Equation (4) in which the value of ρ^* will be replaced by the Q-value for some state-action pair that will be fixed at the beginning. This state-action pair can be any state-action pair in the system and will be called a distinguished state-action pair. The steps in the DP algorithm will be as follows.

Step 1. Set the number of iterations, k, to 0. Set all Q-values to 0, i.e., for all $i \in \mathscr{S}$ and all $a \in \mathscr{A}(i)$, set $Q^k(i,a) = 0$. Set ε to a small positive value. Set α, the step size, to a small positive value less than 1. Select any state-action pair, and call it the distinguished state-action pair, (i^*, a^*).

Step 2. Update for each $i \in \mathscr{S}$ and each $a \in \mathscr{A}(i)$, update $Q^k(i,a)$ as follows:

$$Q^{k+1}(i,a) = (1-\alpha)Q^k(i,a) + \alpha \sum_{j \in \mathscr{S}} p(i,a,j) \left[r(i,a,j) - Q^k(i^*,a^*)\bar{t}(i,a,j) + \max_{b \in \mathscr{A}(j)} Q^k(j,b) \right].$$

Step 3. For each $i \in \mathscr{S}$, compute $v^k(i) = \max_{a \in \mathscr{A}(i)} Q^k(i,a)$ and $v^{k+1}(i) = \max_{a \in \mathscr{A}(i)} Q^{k+1}(i,a)$. Then check if the following norm is less than ε:

$$||v^{k+1} - v^k||_\infty < \varepsilon \tag{5}$$

where v^k denotes the vector of values in the kth iteration and v^{k+1} denotes the same in the $(k+1)$th iteration. If the condition in (5) above is true, go to Step 4; otherwise increase k by 1 and return to Step 2.

Step 4. Compute the ε-optimal policy, μ_ε, as follows: $\mu_\varepsilon(i) = \arg\max_{a \in \mathscr{A}(i)} Q^k(i,a)$ for every $i \in \mathscr{S}$, and stop.

While we do not analyze the algorithm for convergence here, we hope to prove in future work when $\lim_{K\to\infty} \sum_{k=1}^K \alpha^k = \infty$ and $0 < \alpha^k < 1$, the update in Step 2 should solve the Bellman equation in Equation (4), and hence

$$\lim_{k\to\infty} Q^k(i^*, a^*) = \rho^*.$$

A proof of this can envisioned along the lines of the ordinary differential equation framework — outlined in Section 4.4 of Bertsekas and Tsitsiklis (1996) exploiting Markov noise for the simulation-based algorithm. Other potential avenues for showing convergence could involve other results e.g., Szepesvari and Littman (1998) and Jaakkola, Jordan, and Singh (1994).

We now turn our attention to an RL algorithm in which we will seek to avoid the transition probabilities. Here, the step-size must decay with k and cannot be a constant, since we will encounter simulation noise. The main steps in the algorithm are provided in Figure 1.

RL ALGORITHM

Step 1: Set $k = 1$ (where k denotes the number of iterations) and initialize $k_{\max}$ to a large number. Initialize $Q^k(i,a)$ for all states $i \in S$ and all $a \in \mathscr{A}(i)$ to 0. Select any state-action pair, and call it the distinguished state-action pair, (i^*, a^*).

Step 2: Start fresh simulation. Let the system state be $i \in S$. Simulate action $a \in \mathscr{A}(i)$ with probability $1/|\mathscr{A}(i)|$. Let the next decision-making state encountered in the simulator be j. Also, let $t(i,a,j)$ be the random transition time (from state i to state j) and let $r(i,a,j)$ be the immediate reward. Then, update $Q^k(i,a)$ using:

$$Q^{k+1}(i,a) = (1-\alpha^k)Q^k(i,a) + \alpha^k \left[r(i,a,j) - Q^k(i^*,a^*)t(i,a,j) + \max_{b \in \mathscr{A}(j)} Q^k(j,b) \right],$$

where α^k denotes the step size, which should be a function of k, e.g.,

$$\alpha^k = \frac{A}{B+k} \text{ in which } A \text{ could be for instance 1000 and } B \text{ could be 2000.}$$

Step 3: Increment k by 1. If $k < k_{\max}$, set current state i to new state j and then return to Step 2; else go to Step 4.

Step 4: Compute $\mu(i) = \arg\max_{a \in \mathscr{A}(i)} Q^k(i,a)$ for every $i \in \mathscr{S}$; declare μ to be the optimal policy and stop.

Figure 1: An RL algorithm for solving average reward SMDPs via relative value iteration.

Note that in the RL algorithm, $t(i,a,j)$ denotes the *random* transition time in the simulator. Further, the step size should satisfy the standard conditions of stochastic approximation (Kushner and Clark 1978):

$$\lim_{K\to\infty}\sum_{k=1}^{K}\alpha^k=\infty;\quad \lim_{K\to\infty}\sum_{k=1}^{K}\left(\alpha^k\right)^2<\infty.$$

4 NUMERICAL RESULTS

We now numerically test our algorithms on small SMDPs with two states and two actions allowed in each state. We studied four problems, named Cases 1 through 4. We first provide the data for Case 1.

Case 1: We present the transition probability, reward and time matrices.

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

$$\mathbf{R}_1=\begin{bmatrix}6 & -5\\ 7 & 12\end{bmatrix};\mathbf{R}_2=\begin{bmatrix}10 & 17\\ -14 & 13\end{bmatrix};$$

$$\mathbf{T}_1=\begin{bmatrix}10 & 5\\ 120 & 60\end{bmatrix};\mathbf{T}_2=\begin{bmatrix}50 & 75\\ 7 & 20\end{bmatrix}.$$

For the DP algorithm, the transition time $\bar{t}(i,a,j)$ is defined via the **T** matrices above. For the RL algorithm, the *random* transition time is defined as: $t(i,a,j)=\bar{t}(i,a,j)+UNIF(-1,1)$, where $UNIF(a,b)$ denotes a uniformly distributed random number between a and b.

Case 2: All the data will be identical to that for Case 1 except for: $r(1,1,2)=5;\ r(2,2,1)=14$.

Case 3: All the data will be identical to that for Case 1 except for: $r(1,2,1)=12;\ \bar{t}(2,1,1)=12$.

Case 4: All the data will be identical to that for Case 1 except for: $r(1,1,1)=16;\ r(1,2,1)=0$.

We used $\varepsilon=0.01$ in the DP algorithm. Also, in each case, for both algorithms, $(i^*,a^*)=(1,1)$. Exhaustive enumeration was used to determine the optimal policy and the optimal average reward ρ^*. For the RL algorithm, we used the step size rule:

$$\alpha^k=\frac{log(k+1)}{k}\qquad \text{for } k\geq 1;$$

we further found that other rules such as $A/(B+k)$ produced a similar behavior. The RL algorithm was run with $k_{\max}=10,000$.

The optimal policy, determined via exhaustive enumeration, was action 1 in state 1 and action 2 in state 2 in every case. Table 1 shows the Q-values produced by the DP and the RL algorithms. It should be clear from the Q-values in the table that both algorithms generate the optimal policy. Further, in each case, $Q(i^*,a^*)$ is very close to ρ^*.

5 CONCLUSIONS

The average reward SMDP is a longstanding problem in Markov decision theory. In general, the simplest approach to solve it has conventionally been to use policy iteration. In order to use value iteration, one must employ a discretization and then perform a relative value iteration, as discussed above. The discretization adds a thick layer to the computational burden of the algorithm, since one must find a suitable value for

Table 1: Q-values for the RL and DP algorithms.

Case #	ρ^*	$Q(1,1)$	$Q(1,2)$	$Q(2,1)$	$Q(2,2)$
Case 1 (DP)	0.4075	0.4074	-9.99	-22.24	2.94
Case 1 (RL)		0.4427	-10.50	-22.20	3.64
Case 2 (DP)	0.7370	0.7369	-27.02	-49.99	2.61
Case 2 (RL)		0.7187	-26.76	-51.09	2.24
Case 3 (DP)	0.4075	0.4073	-8.18	-4.66	2.94
Case 3 (RL)		0.4439	-8.8566	-4.37	3.68
Case 4 (DP)	0.6098	0.6097	-31.17	-49.60	-14.44
Case 4 (RL)		0.7389	-32.79	-48.83	-12.55

χ. Here, our goal was to consider a more direct approach inspired by Q-Learning. We presented a DP algorithm that requires transition probabilities, but is step-size-based. The step size in the DP algorithm can be constant and very close to 1, potentially leading to swift convergence. Of course, another motivation for developing a step-size-based algorithm was that it has a simulation-based counterpart which can be used to solve the problem in absence of the transition probabilities, i.e., via RL. The latter allows one to solve complex problems the transition probabilities of which are difficult to find but simulators are available. Existing RL algorithms for this problem require an additional update of an iterate. The algorithm presented here avoids the additional iterate and is based on the concept of relative value iteration, first discussed in White (1963).

Significant additional work needs to be carried out in this area. First, convergence proofs need to be developed for both the RL and the DP algorithms. Second, the algorithms need to be tested on larger problems. A number of operations management applications of RL have been covered in Bertsekas and Tsitsiklis (1996), among other places. The text of Sutton and Barto (1998) discusses applications within artificial intelligence. Recently, RL algorithms have also been used in aviation engineering (see e.g., Ng et al. (2004) and Abbeel et al. (2009)); it appears, thus, that our algorithm can be tested on a variety of large-scale engineering problems.

REFERENCES

Abbeel, P., A. Coates, T. Hunter, and A. Ng. 2009. “Autonomous Autorotation of an RC Helicopter”. *Experimental Robotics: Eleventh International Symposium Series* 54:385–394.

Abounadi, J., D. Bertsekas, and V. Sorkar. 2001. “Learning Algorithms For Markov Decision Processes with average cost”. *SIAM Journal of Control and Optimization* 40(3):681–698.

Bertsekas, D. P. 2012. *Dynamic Programming and Optimal Control: Volumes I and II*. Belmont, Massachusetts: Athena Scientific.

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

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

Das, T. K., A. Gosavi, S. Mahadevan, and N. Marchalleck. 1999. “Solving Semi-Markov Decision Problems Using Average Reward Reinforcement Learning”. *Management Science* 45(4):560–574.

Gosavi, A. 2003. *Simulation-Based Optimization: Parametric Optimization Techniques and Reinforcement Learning*. NY: Springer.

Gosavi, A. 2004. “Reinforcement Learning for long-run average cost”. *European Journal of Operational Research* 155:654–674.

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

Jaakkola, T., M. Jordan, and S. Singh. 1994. “On the convergence of stochastic iterative dynamic programming algorithms”. *Neural Computation* 6(6):1185–1201.

Kushner, H., and D. Clark. 1978. *Stochastic Approximation Methods for Constrained and Unconstrained Systems*. New York: Springer Verlag.

Ng, A. Y., H. J. Kim, M. I. Jordan, and S. Sastry. 2004. “Autonomous helicopter flight via reinforcement learning”. In *Advances in Neural Information Processing Systems 17*: MIT Press.

Puterman, M. 1994. *Markov Decision Processes*. Wiley Interscience, New York, USA.

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

Szepesvari, C., and M. Littman. 1998. “A Unified Analysis of Value-Function-Based Reinforcement-Learning Algorithms”. *Neural Computation* 11:11–8.

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

White, D. J. 1963. “Dynamic programming, Markov chains, and the method of successive approximations”. *J. Math. Anal. Appl.* 6:373–376.

AUTHOR BIOGRAPHY

ABHIJIT GOSAVI joined Missouri University of Science and Technology as an Assistant Professor in the Department of Engineering Management and Systems Engineering in 2008. His research interests lie in Markov decision processes, simulation-based optimization, revenue management, and total productive maintenance. He is a member of INFORMS, IIE, ASEE, and POMS. He serves as an Associate Editor for the *Engineering Management Journal* and has served as the Program Chair and as the Division Chair of the Industrial Engineering Division of ASEE. His email address is gosavia@mst.edu and his web page is http://web.mst.edu/~gosavia/.