A simulation-based learning automata framework for solving semi-Markov decision problems under long-run average reward

ABHIJIT GOSAVI1, TAPAS K. DAS2 and SUDEEP SARKAR3

1Department of Industrial Engineering, State University of New York at Buffalo, Buffalo, NY 14260, USA
E-mail: gosavi@eng.buffalo.edu
2Department of Industrial and Management Systems Engineering, and 3Department of Computer Science,
University of South Florida, Tampa, FL 33620, USA
E-mail: das@eng.usf.edu or sarkar@bigpine.csee.usf.edu

Received December 1999 and accepted June 2003

Many problems of sequential decision making under uncertainty, whose underlying probabilistic structure has a Markov chain, can be set up as Markov Decision Problems (MDPs). However, when their underlying transition mechanism cannot be characterized by the Markov chain alone, the problems may be set up as Semi-Markov Decision Problems (SMDPs). The framework of dynamic programming has been used extensively in the literature to solve such problems. An alternative framework that exists in the literature is that of the Learning Automata (LA). This framework can be combined with simulation to develop convergent LA algorithms for solving MDPs under long-run cost (or reward). A very attractive feature of this framework is that it avoids a major stumbling block of dynamic programming; that of having to compute the one-step transition probability matrices of the Markov chain for every possible action of the decision-making process. In this paper, we extend this framework to the more general SMDP. We also present numerical results on a case study from the domain of preventive maintenance in which the decision-making problem is modeled as a SMDP. An algorithm based on LA theory is employed, which may be implemented in a simulator as a solution method. It produces satisfactory results in all the numerical examples studied.

1. Introduction

A large class of problems of sequential decision making under uncertainty, whose underlying probability structure has a controlled Markov chain, can be formulated as Markov Decision Problems (MDPs). The objective underlying the study of these problems is to find the optimal action (control) in each state of the Markov chain. Optimality of actions is usually measured with respect to some performance metric such as the total discounted reward over an infinite time horizon, or the average reward over an infinite time horizon (also called the long-run average reward). In such a characterization (i.e., MDP), the time spent in the transition from one state to another is either unity or else it is ignored (and assumed to be unity) in the calculation of the performance metric. However, problems in which we cannot ignore the transition times (which may be deterministic or random variables) can often be solved using the semi-Markov decision model. This paper is centered on a simulation-based technique related to the semi-Markov model.

The framework of Dynamic Programming (DP) has been used extensively in the literature to solve Markov and semi-Markov decision problems (Puterman, 1994). A stochastic-approximation method called reinforcement learning, which is also based on the framework of DP (see Bertsekas and Tsitsiklis (1996) and Sutton and Barto (1998) for textbook treatments), has also appeared in the literature in recent years.

The theory of Learning Automata (LA) (Narendra and Thatachar, 1989) forms an alternative framework for solving MDPs; a framework that is very distinct from that of DP. It is rooted in the principles of game theory. The LA framework, which is simulation-based, can be used to develop convergent algorithms to solve MDPs under long-run cost (or reward) (Wheeler and Narendra, 1986). A very attractive feature of this framework, unlike traditional DP, is its ability to generate near-optimal solutions without having to compute the one-step transition probability matrices of the decision-making problem. In this paper, we present an extension of the LA framework to the more general Semi-Markov Decision Problem (SMDP). We also conduct
numerical tests on a decision-making problem given in a paper by Das and Sarkar (1999). Their paper contains results for the optimal policy for Preventive Maintenance (PM) of a production-inventory system. They obtain the optimal policy via the harder route, which involves setting up the analytical expressions for the one-step transition probabilities and the steady-state probabilities of the semi-Markov process. (Such analytical techniques are feasible only for small problems. For real world large-sized problems, we have to rely on other techniques such as the one presented here.)

We model the PM optimization problem as a SMDP as in their paper, but use an algorithm based on LA theory as a solution method. We refer to the algorithm as the MCAT (short for Markov Chain Automata Theory) algorithm. The algorithm is shown to produce optimal results in all the numerical examples studied. It was encoded in Arena (Arena is marketed by Rockwell Software) and C. The computational savings obtained from the LA approach are substantial because the LA approach does not compute the transition probabilities and the steady-state probabilities.

The LA approach is a simulation-based optimization technique and as such requires a simulator of the system. The optimal policy, using the LA approach, is “learned” from one simulation run of the system. The LA simulation run is not anything like what is understood by a standard simulation replication, in which the policy is fixed at the start, once and for all, and the system is simulated over a fixed time horizon to estimate the performance metric. Rather, in a LA simulation run, the simulation starts with a random policy, but the policy changes at every decision epoch and the run ends with the system having “learned” the optimal (deterministic) policy. It may be noted here that to simulate the system, it is not necessary to generate the transition probabilities; all one needs is the knowledge of the distributions of the random variables that govern the system behavior.

There is a vast body of literature on simulation-based optimization techniques. Lately, research in simulation-based optimization has received a great deal of attention perhaps because analytical approaches do not fare well on many large, real world, stochastic problems. The Response Surface Method (RSM) was one of the first simulation-based optimization approaches to have appeared in the literature. Applying this approach to our problem would require the simulation of several policies followed by the use of a function-approximation technique, such as regression, to approximate the cost function over the policy space (Law and Kelton, 2000). The RSM will be time consuming for our problem but is nevertheless a robust procedure; which is why it is still used widely.

Gradient-based techniques for simulation optimization compute the gradient of the cost function and use a gradient descent approach. Initial gradient-based methods hinged on the technique of finite-differences (Azadivar and Talavage, 1980). With the simultaneous perturbation technique, Spall (1992) has shown that the gradient-based approach can be made much faster and can be performed with a modest number of function evaluations. The number of function evaluations is an important issue in simulation-based optimization because each function evaluation can take up a considerable amount of computer time. The literature speaks of several other approaches that can be combined with simulation; especially when the problem is one of finding the optimal values of a set of parameters to minimize a cost function of the parameters in question. This problem is often called a problem of parametric optimization as opposed to control optimization. In the latter, one seeks to find the optimal action in each state visited by the relevant stochastic process. Although the problem considered in this paper is one of control optimization, such problems can be often solved by using a parametric-optimization approach especially when the optimal policy has a threshold nature. Therefore, some discussion on the more recent approaches, advocated by researchers for solving the parametric simulation-optimization problem, is in order.

Metaheuristics such as tabu search (Glover, 1990), simulated annealing (Homem-de-Mello, 2001), and genetic algorithms (Boesel et al., 2003) have been combined with simulation to solve problems of discrete stochastic optimization, and a varying degree of success has been reported. A genetic algorithm has been used in the optimization module of a commercial software (Harrell et al., 2000). Random search methods proposed by Andradottir (1995, 1996) have been shown to be convergent. Ranking and Selection (R & S) procedures (Goldsmann et al., 1991; Nelson, 1992; Goldsmann and Nelson, 1994) are statistically proven methods for comparing alternative solutions (systems or configurations) and selecting the best with a high probability. Jacobson and Schruben (1999) present a methodology to compute the gradient of the objective function and Hessian estimators using harmonic analysis. They use Taylor series to approximate the objective function and show that these estimates can be made from just one replication of the system. In this respect their work bears resemblance to MCAT; however, the problem they attack is one of parametric optimization. We have more to say on these topics (see Section 5.2) in the context of the case study used in this paper. For some other references to these methods, see Rubinstein and Shapiro (1983), Glasserman (1991) and Bonnans and Shapiro (2000). A comprehensive source is Andradottir (1998).

This paper, to the best of our knowledge, is the first to use a LA approach to solve a SMDP. Previous computational work on the use of LA to problems of control optimization was limited to small MDPs with a handful of states (Wheeler and Narendra, 1986).

The rest of this paper is organized as follows. Section 2 contains an overview of the existing literature on SMDPs and automata theory. Section 3 describes the SMDP. Section 4 presents an overview of the automata framework. Section 5 contains the details of the LA algorithm used in this paper. Section 6 is centered on the PM case study. It also
discusses results from our application of the LA algorithm. Section 7 concludes this paper.

2. Overview of related literature

The existing literature reveals that Markovian and semi-Markovian models have been studied extensively for decision-making purposes and have been used in the context of DP. For a detailed account on MDPs/SMDPs, the reader is referred to Puterman (1994). Applications of these models can be found in a multitude of stochastic optimization problems, some examples of which are: finding optimal inspection policies in a quality control problem (Cassandras and Han, 1992), optimal control of a queueing problem with two customer types (Shiroyama, 1991), optimal maintenance in random environments (Ozeki, 1995), part selection in an FMS (Seidmann and Schweitzer, 1983), optimal maintenance of a production-inventory system (Das and Sarkar, 1999), and a stochastic economic lot-size scheduling problem with two part types (Bai et al., 1996). Sennott (1999) discusses examples of decision making in stochastic systems, especially those in queueing systems.

The theory of learning in controlled Markov chains, in the context of game theory, has received an excellent treatment in Narendra and Thathachar (1989). The first significant result, which may be applied to MDPs, appeared in Wheeler and Narendra (1986). Some other comprehensive references to research related to learning algorithms, using the paradigm of game theory, are Lakshmivarahan (1981) and Lakshmivarahan and Narendra (1981). Das and Sarkar (1999) derive the optimal policy for the PM problem using a semi-Markov model and hence we use their problem to benchmark the performance of the MCAT algorithm.

3. SMDPs

A SMDP, which has an underlying Markov Chain (MC), can be characterized by the following five elements: (i) the state space of the MC; (ii) the action (decision or control) space; (iii) the transition cost (or reward) matrix; (iv) the transition probability matrix; (v) the transition time matrix. All the states in the MC are not necessarily decision-making states. This does not mean that we are dealing with an uncontrolled MC. Rather it means that there are some states in the MC in which there is no decision making involved and only one action is taken.

The two stochastic processes that accompany the underlying MC are; (i) the Natural Process (NP); and (ii) the Decision-Making Process (DMP). The NP tracks all the state changes in the MC while the DMP tracks only the decision-making states. Obviously, the NP and the DMP coincide at the decision-making instants. Any given transition in the SMDP is characterized by a current state, the next state and the action that caused it. The transition probability refers to the probability that the DMP goes from the given current state to the given next state, under the given action. Similarly, the transition cost and the transition time denote the cost incurred and the time spent, respectively, in any given transition. The time spent in a transition for a SMDP is usually a random variable drawn from a general distribution. Next, we quickly formalize the above discussion.

Let \( X_m \) and \( T_m \) denote the system state and time respectively at the \( m \)th decision-making instant. We define two stochastic processes: \( X = \{ X_m : m \in \mathbb{N} \} \) and \( T = \{ T_m : m \in \mathbb{N} \} \) where \( \mathbb{N} \) is the set of natural numbers. We also define a joint process \( (X, T) = \{(X_m, T_m) : m \in \mathbb{N}\} \), which is the DMP underlying the Markov chain, \( X \). If it can be shown that

\[
P[X_{m+1} = j, T_{m+1} = t | X_0, \ldots, X_m, T_0, \ldots, T_m] = P[X_{m+1} = j, T_{m+1} - T_m \leq t | X_m, T_m],
\]

then the DMP \( (X, T) \) is a Markov renewal process. If \( Y_i \) denotes the state of the system at any time \( t \), then in light of the \( (X, T) \) process being a Markov renewal process, the stochastic process \( Y = \{ Y_t : t > 0 \} \) will be a SMDP. The process, \( Y \), is also the NP of the MC. The average reward of a SMDP, starting at state \( i \) and continuing with policy \( \pi \) for an infinite time period can be given (using the renewal reward theorem; cf. Puterman (1994) and Ross (1997)) as:

\[
\rho^\pi(i) = \lim_{N \to \infty} \left( 1/N \right) E^\pi_i \left[ \sum_{k=1}^N g(i_k, a_k, i_{k+1}) \right].
\]

where \( i_k \) and \( a_k \) denote the state and action, respectively, at the \( k \)th decision-making epoch, \( g(i_k, a_k, i_{k+1}) \) denotes the immediate reward earned in the transition from state \( i_k \) to state \( i_{k+1} \) under the action \( a_k \) taken in state \( i_k \), \( t(i_k, a_k, i_{k+1}) \) denotes the time taken in the same transition, and \( E^\pi_i[\cdot] \) denotes the expectation operator under policy \( \pi \) when the initial state is \( i \). If the MC \( X \) has a unichain transition probability matrix (see Puterman (1994) for a detailed analysis and definition of a unichain transition probability matrix), the expected average cost does not vary with the initial state for any stationary policy.

In the next section, we give a brief outline of the theory of LA for MDPs.

4. A simulation-based LA framework

The idea underlying any learning algorithm for MDPs/SMDPs is simple and intuitively appealing (see Fig. 1). The learning agent (or decision maker) starts with a random policy. As it implements the policy (either in real time or in simulation), it obtains a response from the system (environment). This response is used as feedback to update the policy. Depending on what feedback the actions of a policy generate, the actions are deemed good or bad. Over time, with trial and error, the system learns.
the optimal action in each state. The feedback or response is actually the immediate reward or cost generated by an action. We next discuss the framework in more formal terms.

Let \( \mathcal{A}_i \) denote the set of actions available at state \( i \). Hence, the union of \( \mathcal{A}_i \) over \( i \) gives the action space. With every state-action pair, we associate a probability \( P(i, a) \) of taking action \( a \) in state \( i \). Clearly, \( \sum_{a \in \mathcal{A}_i} P(i, a) = 1 \). At the start of the learning process, the policy is random and each action is equally likely. Hence, in the beginning, \( P(i, a) = 1/|\mathcal{A}_i| \) is the number of possible actions in state \( i \). If the performance of an action is good (in terms of the costs or rewards), the probability of that action is increased and if the performance is poor, the probability is reduced. This is called updating of the state-action probabilities. Of course, the updating scheme must always ensure that the sum of the probabilities of the different actions for any given state is always one.

The feedback mechanism works as follows. Whenever a state \( i \) is revisited, the average reward earned in the system since its last visit to that state is used as a measure of the response. The average reward is calculated by the total reward earned since the last visit to that state divided by the number of decision-making state transitions since the last visit to that state. Thus, if state \( i \) is visited, the response \( \psi(i) \) in the MDP is:

\[
\psi(i) = \frac{R(i)}{N(i)},
\]

where \( R(i) \) is the total reward earned since the last visit to state \( i \) and \( N(i) \) is the number of transitions that have taken place in the DMP since the last visit to \( i \). (The SMDP case is discussed in the next section.) The response is further normalized to ensure that it lies between zero and one; since it is used to update the probabilities, which must lie in the same range. The normalized response is called feedback. The normalization is done using:

\[
\beta(i) = \frac{\psi(i) - \psi_{\text{min}}}{\psi_{\text{max}} - \psi_{\text{min}}},
\]

where \( \psi_{\text{min}} \) denotes the minimum response possible in the system and \( \psi_{\text{max}} \) denotes the maximum response possible in the system. The feedback is used to update the probabilities. The conversion of the response to feedback is a research topic by itself, and for further reading, see Narendra and Thatachar (1989).

Figure 2 depicts a three-state two-action SMDP and shows how the MCAT algorithm works in a simulator. Here \( \mathcal{A}_1 = \mathcal{A}_2 = \mathcal{A}_3 \). We next discuss a learning scheme for updating the action probabilities.

### 4.1. Learning scheme

The scheme for updating the probabilities using feedback is also a topic of considerable research and several schemes have been suggested in the literature. The scheme that gave us the best results is known as the reward-in-action scheme (Narendra and Thatachar, 1989). In what follows, we describe this scheme.

When the system visits a new state \( i \), the probabilities of all the actions in state \( i \) are first updated. For this, \( \beta(i) \) is calculated first, and then the probabilities are updated as shown in Equation (1). Let \( A(i) \) denote the action selected in state \( i \in S \) in its last visit to \( i \), and let \( \eta \) denote the “learning rate.” Update \( P(i, u) \) as shown next:

\[
P(i, u) \leftarrow P(i, u) + \eta \beta(i) I(A(i) = u) - \eta \beta(i) P(i, u),
\]

where \( u \) denotes the action whose probability of being selected in state \( i \) is to be updated, and

\[
I(\cdot) = \begin{cases} 
1 & \text{if the condition inside the round brackets is satisfied,} \\
0 & \text{otherwise.} 
\end{cases}
\]

The reason why the above scheme is so named can be explained as follows. An action that results in a good reward automatically has a high value for \( \beta \) and thus significantly increases the probability of that action. On the other hand, an action that results in a poor reward has a low value for \( \beta \) and thus does not significantly change the probabilities (see Equation (1)). Thus, a reward is acted upon, but a poor response is somewhat ignored (inaction). As the probabilities are updated, usually one action starts predominating. In other words, the probability of the predominant action starts converging to one, while the probabilities of the other actions start converging to zero. If there are \( k \) optimal actions, then the probability of each of the optimal actions starts converging to \( 1/k \). That such a mechanism will converge to the optimal solution has been established by Wheeler and Narendra (1986).
A straightforward extension of this MDP framework to the SMDP case is possible. The response term, $\psi(i)$, for a given state $i$, requires a simple modification. The response term actually denotes the average reward earned since the last visit to the state. So instead of dividing the total earned reward by the number of transitions (since the last visit) in the DMP of the SMDP, we divide the total reward by the total time spent in the NP of the SMDP since the last visit. Thus, the response for the SMDP may be calculated as:

$$\psi(i) = \frac{R(i)}{T(i)},$$

where $T(i)$ denotes the time spent in the NP of the MC since the last visit to the state $i$. All other steps of the updating process for a SMDP will be identical to those followed for a MDP. Next, we present a step-by-step description of the MCAT algorithm, and in Section 4.3 we illustrate the use of the MCAT algorithm on a simple three-state SMDP.

### 4.2. The MCAT algorithm

A step-by-step description of the MCAT algorithm is now presented.

**MCAT algorithm**

A. Let $S$ denote the set of decision-making states in the SMDP. Set iteration count $m = 0$. Initialize action probabilities $P(i, a) = 1/r_i$ for all $i \in S$ and $a \in A_i$. Set the cumulative reward $C_i(t) = 0$ and the cumulative time $C_t(t) = 0$ for all $i \in S$. Also initialize to zero the total reward earned in system from the start, $R_{current}$ and the simulation clock, $T_{current}$. Initialize $s_{\min}$ and $s_{\max}$ as given in the text. Pre-fix $u_i$ to any action that is allowed in state $i$. Start system simulation.

B. While $m < \text{MAX\_STEPS}$ do

If the system state at iteration $m$ is $i \in S$,

**Step 1.** If state $i$ has been visited for the first time, go to Step 3; else, in the following order, set:

$$R(i) \leftarrow R_{current} - C_i(t), T(i) \leftarrow T_{current} - C_t(t),$$

$$\psi(i) \leftarrow \frac{R(i)}{T(i)}$$

and finally $\beta(i) \leftarrow \frac{\psi(i) - \psi_{\min}}{\psi_{\max} - \psi_{\min}}$.

**Step 2.** Update $P(i, u_i)$ using Equation (1).

**Step 3.** With probability $P(i, a)$, select an action $a \in A_i$.

**Step 4.** Set $A(i) \leftarrow a, C_i(t) \leftarrow R_{current}$, and $C_t(t) \leftarrow T_{current}$.

**Step 5.** Simulate the chosen action $a$. Let the system state at the next decision epoch be $j$. Also let $t(i, a, j)$ be the transition time, and $g(i, a, j)$ be the immediate reward earned in the transition resulting from selecting action $a$ in state $i$.

**Step 6.** Set $R_{current} \leftarrow R_{current} + g(i, a, j)$ and $T_{current} \leftarrow T_{current} + t(i, a, j)$.

**Step 7.** Set current state $i$ to new state $j$, and $m \leftarrow m + 1$.

The algorithm presented in this paper is for a PM case study where there are two possible actions in each state. The terms $\psi_{\min}$ and $\psi_{\max}$ denote the minimum and maximum values, respectively, of the average reward from the system over the policy space. Knowledge of $\psi_{\max}$ and $\psi_{\min}$ can be obtained easily from the values of the cost elements in the problem and some elementary extreme-case analysis. (See Section 5.1 for further details.)
There are a number of ways to terminate the algorithm. One way is to wait until all probabilities have approached zero or one. The other method is to run the algorithm for a finite number of iterations and then choose the most probable action in each state.

4.3. An illustrative three-state example

In this section, we show how MCAT works on a simple three-state SMDP. For the sake of simplicity, we shall first assume that each state has two possible actions. In the next section, we shall describe how MCAT is combined with a simulator, and then in the following section, we show the details of how the action probabilities are updated.

4.3.1. MCAT and the simulator

We will not require the values of the transition probabilities of the underlying MC; a simulator of the system will be sufficient. Using the simulator, we can generate a trajectory of states and when the simulator visits a given state certain quantities related to that state will be updated.

Let us assume that the system starts out in state $i$. With a probability of $P(i, a)$, where $a \in A_i$, action $a$ is chosen. The updating required is performed within the simulator and the action chosen is simulated. Let the next state be $j$. Again, an action is chosen, the quantities in question are updated and the selected action is simulated. This continues until, in each state, the probability of one action(s) starts dominating the probabilities of the other actions. We next illustrate the updating process with numerical values.

4.3.2. MCAT and the updating

Let us assume that the first few states of the trajectory generated by the simulator turn out to be $(2, 1, 3, 1)$. We begin by setting $C_i(i) = 0$ and $C_i(i) = 0, \forall i$. We also set $R_{\text{current}}$ and $T_{\text{current}}$ to zero. Also, we shall assume that $u_i = 1, \forall i$, $\psi_{\text{max}} = 10$, $\psi_{\text{min}} = -5$ and $\eta = 0.1$. Also:

$$P(1, 1) = P(1, 2) = P(2, 1) = P(2, 2) = P(3, 1) = P(3, 2) = 0.5.$$ 

The updating calculations performed in each state are listed in Table 1. The updating is continued in this fashion for a large number of iterations. The most likely action (the action with the largest probability) in each state is considered to be the best action for that state.

We next discuss convergence of the MCAT for SMDPs. Since the convergence result is based on game-theoretic concepts, we will present some key game-theoretic ideas to relate the SMDP to a stochastic game.

4.4. Semi-Markov convergence

A SMDP can be thought of as a non-zero sum game with multiple players and a common payoff. We associate one player (automaton) with each state in the MC. Then running the system that has $N$ decision-making states with a fixed policy is equivalent to playing a game with $N$ players. The common payoff to all the players is defined in the SMDP by the long-run average reward associated with the policy. Our task then is to find the equilibrium point of the game; where the payoff is maximized. We are measuring the payoff in terms of the feedback from the system, i.e., $\beta$. Nash (1950) proved that all finite games have at least one pure or mixed strategy equilibrium.

The convergence of the MCAT algorithm has been established via a game-theoretic connection (Wheeler and Narendra, 1986) in a very general setting that includes the MDP case. We need to define two terms before stating the result.

Let us define $\beta^k(i)$ to be the feedback received by the automaton in the $i$th state at the $k$th iteration of the algorithm and $M^k(i)$ to be its expected value. The feedback is associated with given values for the vector $P^k = (P^k(i, 1), P^k(i, 2), \ldots, P^k(i, r))$, where $P^k(i, a)$ denotes the probability of selecting action $a$ in state $i$ in the $k$th iteration of the algorithm. A mathematical definition for $M^k(i)$ is:

$$M^k(i) \equiv E[\beta^k(i) \mid P^k].$$

We next define $d(\pi)$ to be the expected feedback received by the MDP or SMDP when a policy $\pi$ is followed. A policy is a rule that associates a unique action with each state. Clearly, in the SMDP (as in the MDP), there is a single performance metric to be optimized (the long-run average reward in this case). It can be shown that:

$$\lim_{k \to \infty} M^k(i),$$

exists (Narendra and Thatchar, 1989) and is the same for all values of $i$. We denote this limit by $M^k$. Then Wheeler and Narendra’s result can be stated as:

**Theorem 1.** Let $G$ be a common-payoff game among $N$ players $(i = 1, 2, \ldots, N)$; the $i$th player has $r_i$ actions. Assume that all the players use the reward-inaction learning scheme. Further, let $G$ have a unique equilibrium $\pi^*$, whose expected feedback is $d(\pi^*)$. Then for any $\varepsilon > 0$, there exists an $\eta^*$, $0 < \eta^* < 1$, such that for any positive $\eta < \eta^*$:

$$\lim_{k \to \infty} E[M^k] > d(\pi^*) - \varepsilon.$$ 

What this result says is that by a suitable choice of the learning rate (or step size), $\eta$, in the reward-inaction scheme, one can obtain a feedback that is arbitrarily close to that associated with the optimal policy. In other words, the reward-inaction schemes are $\varepsilon$-optimal.

The above-stated result holds for any scenario (as long as the conditions of the learning algorithm are satisfied) in which one seeks to maximize the feedback. According to the normalization scheme, a feedback equaling one is the best, as it is associated with $\psi_{\text{max}}$ and that associated with
Table 1. Sample calculations from a three-state SMDP. Iteration 4 updates probability $P(1, 1)$ from 0.5 to 0.476, via Equation (1), using $R(1) = 3.5$, $T(1) = 1.64$, $\psi(1) = 2.134$ and $\beta(1) = 0.476$

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Current state ($i$), next state ($j$)</th>
<th>Action, $a$, in $i$</th>
<th>$g(i,a,j)$, $t(i,a,j)$</th>
<th>$R_{current}$, $T_{current}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>[2, 1]</td>
<td>1</td>
<td>[0, 0]</td>
<td>[4.5, 2.34]</td>
</tr>
<tr>
<td>2</td>
<td>[1, 3]</td>
<td>2</td>
<td>[4.5, 2.34]</td>
<td>[3.5, 0.11]</td>
</tr>
<tr>
<td>3</td>
<td>[3, 1]</td>
<td>2</td>
<td>[8, 2.45]</td>
<td>[−1, 1.55]</td>
</tr>
<tr>
<td>4</td>
<td>[1, 2]</td>
<td>1</td>
<td>[7, 4]</td>
<td>[2.4, 1.23]</td>
</tr>
</tbody>
</table>

zero is the worst; hence the term “maximize.” Therefore, for the SMDP, this result will hold as long as the feedback is calculated (properly), keeping track of the time spent in earning any given reward. For the SMDP, one is interested in maximizing the average reward calculated per unit time over an infinitely long time horizon. (This differs from the MDP case, where the performance metric is calculated on a unit-step basis). And therefore, we must divide the incremental reward $R(i)$ by the incremental time $T(i)$. This is how we justify Equation (2). Our equation captures the performance metric we are interested in.

Of course, the result will hold regardless of how we define the response, as long as it is normalized, but the policy learned will only be as good as our definition of the performance metric. In other words, all that the result guarantees us is that the policy learned will optimize the performance metric chosen. Because of the generality of its conditions, the result actually holds even when the system does not have an underlying MC (Wheeler and Narendra, 1986). Furthermore, it is likely to have applications to problems modeled with partially observable MCs.

5. Case study

To demonstrate empirically that the MCAT algorithm generates a near-optimal policy on SMDPs from real-life problems, we choose a case study from the PM literature (Das and Sarkar, 1999). This case study is especially suitable because the optimal policies are available in Das and Sarkar (1999) on some experiments; and hence it can serve as a suitable vehicle for benchmarking MCAT. We now describe the problem in some detail.

Consider a production-inventory system as shown in Fig. 3. It produces a single product type to satisfy an external demand process. The system is prone to failures; the time between failures, a random variable, is not exponentially distributed. Hence, PM (Lewis, 1994) is an option to reduce the downtime of the machine. Demands that arrive when the inventory buffer is empty are not backordered and are therefore lost. A finished-product inventory buffer of the product is maintained so as to minimize the chances of losing demand when the machine is down due to breakdown or maintenance. The system stops production.
when the inventory reaches $Z_1$ and the production resumes when the inventory drops to $Z_2$. During its vacation, i.e., when production stops due to the inventory reaching its upper limit $Z_1$, the system is assumed not to age or fail. PM decisions are made only at the completion of a part production. The problem of finding the optimal PM decisions can be cast as a SMDP. We next discuss the SMDP formulation.

An infinite supply of raw material is assumed. When the system produces a part, the part goes into the inventory buffer. When a demand arrives, the buffer is depleted by one unit; if the buffer is empty that particular demand goes away never to return. As the machine ages during production, it can fail. The repair time and the production times are random variables. After the end of the repair or maintenance, the machine is assumed to be as good as new. When the machine completes the manufacture of one part, two options are available to the decision maker: (i) to produce one more part; and (ii) to schedule a maintenance. Clearly, when the decision maker is faced with the decision of choosing between one more production cycle and a maintenance, (s)he has to make the decision based on the age of the machine and the number of parts in the buffer ($w$). A measure of the age is the number of parts produced since the last repair or maintenance. This will be denoted by $c$ (also called the production count). The decision-making state space of the system may hence be denoted by $(w, c)$. For more details on the semi-Markov model and the method used to obtain the optimal policies, the reader is referred to Das and Sarkar (1999).

### 5.1. Numerical results

We present results from numerical examples of two types for which: (i) theoretical results via exact MC analysis are available; and (ii) optimal solutions are not available. It is usually the case that a few modeling complications or an increase in the size of the state space for stochastic systems can cause analytical methods to break down; then one has to turn to simulation for modeling the system. It is not surprising that the MC approach of Das and Sarkar (1999) breaks down if the buffer limits $(Z_1, Z_2)$ are increased from $(3, 2)$ (an analytically-tractable size) to vectors of larger entries. It is for these cases (type 2) that we turn to a popular heuristic based on renewal theory from the domain of PM. The idea is again to benchmark MCAT's performance against some standard: the optimal solution or a heuristic solution. The renewal-theory heuristic that we have used is known as the Age-Replacement (AR) heuristic. Details related to the heuristic have been placed in the Appendix.

The demand arrival process (with batch size of one) is Poisson ($\gamma$). The time between machine failures, the production time and the machine repair time are gamma distributed with parameters $(k, \mu)$, $(d, \lambda)$, $(r, \delta)$, and respectively. The time for PM has a uniform distribution with parameters $(a, b)$. For each of the 14 parameter sets (shown in Table 2), the cost parameters considered are denoted by: $C_d$ (net revenue per demand serviced), $C_r$ (cost per repair) and $C_m$ (cost per maintenance). The results (average reward) for the single-product inventory system obtained from the theoretical model/heuristic model, and those obtained from the LA algorithm are summarized in Table 3. In the first nine cases, the LA results are compared to the optimal results and in the last five cases, the comparisons are with AR.

The first nine cases have a smaller state space compared to the last five cases. Exact expressions for the transition probabilities (in the first nine cases) were derived by Das and Sarkar (1999) for the buffer limits $(Z_1, Z_2) = (3, 2)$. However, the expressions were derived for these numbers specifically; new values would require a re-determination of the expressions. Furthermore, a unit increase in the upper buffer limit needs at least 25 additional expressions. (The number 25 is the minimum value of $c$ at which the machine

<table>
<thead>
<tr>
<th>System</th>
<th>Time between demands ($\gamma$)</th>
<th>Time between failure ($k, \mu$)</th>
<th>Time between product ($d, \lambda$)</th>
<th>Maintenance time ($a, b$)</th>
<th>Repair time ($r, \delta$)</th>
<th>Inventory limits ($Z_1, Z_2$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1/10</td>
<td>(8, 0.08)</td>
<td>(8, 0.8)</td>
<td>(5, 20)</td>
<td>(2, 0.01)</td>
<td>(3, 2)</td>
</tr>
<tr>
<td>2</td>
<td>1/10</td>
<td>(8, 0.008)</td>
<td>(8, 0.8)</td>
<td>(5, 20)</td>
<td>(2, 0.01)</td>
<td>(3, 2)</td>
</tr>
<tr>
<td>3</td>
<td>1/7</td>
<td>(8, 0.08)</td>
<td>(8, 0.8)</td>
<td>(5, 20)</td>
<td>(2, 0.01)</td>
<td>(3, 2)</td>
</tr>
<tr>
<td>4</td>
<td>1/15</td>
<td>(8, 0.08)</td>
<td>(8, 0.8)</td>
<td>(5, 20)</td>
<td>(2, 0.01)</td>
<td>(3, 2)</td>
</tr>
<tr>
<td>5</td>
<td>1/15</td>
<td>(8, 0.08)</td>
<td>(8, 0.8)</td>
<td>(25, 40)</td>
<td>(2, 0.01)</td>
<td>(3, 2)</td>
</tr>
<tr>
<td>6</td>
<td>1/15</td>
<td>(8, 0.08)</td>
<td>(8, 0.8)</td>
<td>(5, 20)</td>
<td>(2, 0.02)</td>
<td>(3, 2)</td>
</tr>
<tr>
<td>7</td>
<td>1/15</td>
<td>(8, 0.08)</td>
<td>(8, 0.8)</td>
<td>(5, 20)</td>
<td>(4, 0.02)</td>
<td>(3, 2)</td>
</tr>
<tr>
<td>8</td>
<td>1/15</td>
<td>(8, 0.01)</td>
<td>(8, 0.8)</td>
<td>(5, 20)</td>
<td>(4, 0.02)</td>
<td>(3, 2)</td>
</tr>
<tr>
<td>9</td>
<td>1/20</td>
<td>(8, 0.04)</td>
<td>(8, 0.4)</td>
<td>(5, 20)</td>
<td>(4, 0.02)</td>
<td>(3, 2)</td>
</tr>
<tr>
<td>10</td>
<td>1/15</td>
<td>(8, 0.08)</td>
<td>(8, 0.8)</td>
<td>(5, 20)</td>
<td>(2, 0.01)</td>
<td>(5, 4)</td>
</tr>
<tr>
<td>11</td>
<td>1/15</td>
<td>(8, 0.08)</td>
<td>(8, 0.8)</td>
<td>(5, 20)</td>
<td>(2, 0.01)</td>
<td>(6, 5)</td>
</tr>
<tr>
<td>12</td>
<td>1/15</td>
<td>(8, 0.08)</td>
<td>(8, 0.8)</td>
<td>(5, 20)</td>
<td>(2, 0.01)</td>
<td>(7, 6)</td>
</tr>
<tr>
<td>13</td>
<td>1/15</td>
<td>(8, 0.08)</td>
<td>(8, 0.8)</td>
<td>(5, 20)</td>
<td>(2, 0.01)</td>
<td>(8, 7)</td>
</tr>
<tr>
<td>14</td>
<td>1/15</td>
<td>(8, 0.08)</td>
<td>(8, 0.8)</td>
<td>(5, 20)</td>
<td>(2, 0.01)</td>
<td>(9, 8)</td>
</tr>
</tbody>
</table>
failsaalmostsurely.)Asuch,webodonthavetheoptimalexampleforthebestfivecases,where$Z_1 > 4$. 

Anapproximationforthevalueof$\psi_{\text{min}}$canbeobtainedfromtheworst-case scenario, inwhichnomaintenanceis performed.Insuchasituation,$\psi_{\text{min}}$canbeapproximatedbythequotientofthenegativereward (cost) of one repair and the mean time between failure. Of course,$\psi_{\text{max}}$is set to some value higher than$\psi_{\text{min}}$. In our study, we know theoptimal solution in some cases and hence the value of$\psi_{\text{max}}$for those cases. When the optimal solution cannot be obtained, a heuristic (such as AR) is often used, and then$\psi_{\text{max}}$canbe set to be a multiple of the average reward obtainedfromtheheuristic.

A statistical test comparing two means (the average reward obtained from the policy generated by the MCAT algorithm and that obtained from the theoretical results of Das and Sarkar) reveals no significant difference at a 95% confidence level for the first nine cases. A similar test does show a significant difference, in the last five cases (between the MCAT and AR values). This shows that MCAT out-performs AR, in the experiments performed here.

We found that the learning rates do affect the solution that the algorithm generates. The learning rate was set to 0.1 during our calculations. We found that lowering the learning rate below 0.1 made the learning process slower, although convergence to the optimum was still achieved. However, when the learning rate was increased above 0.1, the algorithm generated a suboptimal solution. The reason for this can be traced to the convergence result which states that the learning rate has to be sufficiently small.

### 5.2. Other simulation optimization methods

The PM case study can also be cast as a stochastic integer program. But there are two difficulties with this. Firstly, it is not easy to find the objective function in closed-form and secondly, the solution space will be enormous for a pointwise evaluation of the search space via simulation. A stochastic integer formulation is shown below.

Maximize $h(x_0, x_1, \ldots, x_{\theta})$ where $x_i \in \{0, 1, 2, \ldots, \theta\}$.

Here $\theta$ denotes the smallest production count at which the machine fails with a probability approaching one, and $h(\cdot)$ denotes the long-run average reward earned by following a policy which maintains the machine when there are $i$ units in the buffer and the production count equals $x_i$. (Recall that $Z_1$ is the upper limit for the buffer.) Hence, it is clear that the total number of policies is equal to $(\theta + 1)^{Z_1}$ for $\theta$. For most of the cases considered in this paper, $\theta = 25$ (Das and Sarkar, 1999), and so the total number of alternative configurations (each configuration is associated with a policy) is $26^4$. Therefore, a separate evaluation of each configuration will be computationally burdensome.

**R&S** procedures are recommended for systems with up to 20 or 30 configurations; however, recently they have been used on larger problems (Boesel et al., 2000). Boesel et al. (2000) have designed software that uses ranking and selection procedures in combination with a genetic algorithm search that has a remarkable statistical error control. In particular, the algorithm search delivers a statistical guarantee of the solution generated after the search method ends. An interesting feature of the MCAT algorithm and the genetic algorithm procedure, as described in Boesel et al. (2000), is that both use a somewhat similar philosophy. Both use selection probabilities that favor the seemingly better “solutions” while retaining the probabilities of the “solutions” that seem to be giving poorer feedback. However, the analogy ends here because a genetic algorithm search is designed for solving a problem of parametric optimization whereas the MCAT search is useful in control optimization.

### 6. Conclusions

This paper presented an application of simulation-based optimization to a hard problem in manufacturing. The MCAT algorithm is representative of a new paradigm in simulation-based optimization. For decision-making problems, it forms an alternative both to traditional response surface and gradient-based methods and to reinforcement learning. The extension of the MCAT framework from the MDP to the SMDP, which is a contribution of this paper, should make it more amenable to real-life problems, which more frequently tend to be SMDPs than MDPs. Also, the theory of LA remains largely unknown in the simulation-optimization community, and one objective of this paper was to show how this useful framework may be utilized in solving complex problems. We hope that this will lead to more applications along similar or related lines to difficult problems of stochastic optimization. We are currently working on the use of Bayesian networks to model the

---

**Table 3.** Results from the MCAT algorithm, which were averaged over 30 runs, where each simulation run lasted for 1000000 time units (note that $C_d = 1$, $C_r = 5$ and $C_m = 2$)

<table>
<thead>
<tr>
<th>System</th>
<th>Optimal/ AR average reward ($/ \text{per unit time}$)</th>
<th>MCAT average reward ($/ \text{per unit time}$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.034</td>
<td>0.034</td>
</tr>
<tr>
<td>2</td>
<td>0.076</td>
<td>0.075</td>
</tr>
<tr>
<td>3</td>
<td>0.035</td>
<td>0.035</td>
</tr>
<tr>
<td>4</td>
<td>0.028</td>
<td>0.028</td>
</tr>
<tr>
<td>5</td>
<td>0.025</td>
<td>0.025</td>
</tr>
<tr>
<td>6</td>
<td>0.031</td>
<td>0.030</td>
</tr>
<tr>
<td>7</td>
<td>0.028</td>
<td>0.028</td>
</tr>
<tr>
<td>8</td>
<td>0.057</td>
<td>0.057</td>
</tr>
<tr>
<td>9</td>
<td>0.020</td>
<td>0.020</td>
</tr>
<tr>
<td>10</td>
<td>0.01531</td>
<td>0.02951</td>
</tr>
<tr>
<td>11</td>
<td>0.01545</td>
<td>0.02993</td>
</tr>
<tr>
<td>12</td>
<td>0.01578</td>
<td>0.02957</td>
</tr>
<tr>
<td>13</td>
<td>0.01607</td>
<td>0.03012</td>
</tr>
<tr>
<td>14</td>
<td>0.01608</td>
<td>0.03063</td>
</tr>
</tbody>
</table>
action probabilities. We are also exploring the possibility of using the framework on the more difficult, partially observable version of the SMDP. Other possible avenues for further research in this area include the study of this framework for discounted rewards.

Acknowledgements

The authors would like to thank the simulation area editor Professor Dave Goldsman and an anonymous reviewer for carefully reading the manuscript and making various suggestions that have improved the quality of this work. This research was partly funded by National Science Foundation via grant ITR/AP 0114007 to the first author and grant DMI-0113946 to the second and the third author.

References


Homem-de-Mello, T. (2001) On the convergence of simulated annealing for discrete stochastic optimization. Working Paper, Department of Industrial, Welding and Systems Engineering, Ohio State University, Columbus, OH 43210, USA.


Appendix

The AR heuristic, which was used for benchmarking MCAT’s performance on large-sized problems, is developed via the powerful renewal-reward model.

With an analytical expression for the cost of maintaining the machine, when its age is \( \tau \), one can use non-linear programming to find the optimal value of the age, \( \tau \), that minimizes the average cost. See Ross (1997) for details of this model. The average cost using the renewal-reward theorem can be written as \( EC / CT \), if \( EC \) is the expected renewal cost and \( CT \) is the expected renewal time. Let \( X \) denote the time between failures. Then:

\[
EC = (1 - F(\tau))C_m + F(\tau)C_f,
\]

where \( \tau \) is the age of maintenance, \( F(x) \) is the cumulative distribution function of the random variable \( X \), \( C_m \) is the cost of one maintenance, and \( C_f \) is the cost of one repair.
Also:

\[ CT = \tau F(\tau) + \int_{\gamma}^{\tau} xf(x)dx + (\tau + \tau_m)(1 - F(\tau)) \]

where \( f(x) \) denotes the probability density function of the random variable \( X \) whereas \( \tau_m \) and \( \tau_r \) denote the mean time for maintenance and failure, respectively.

Biographies

Abhijit Gosavi has been an Assistant Professor at the State University of New York at Buffalo since May 2003. He has a PhD in Industrial Engineering (University of South Florida, 1999) and a BS (Jadavpur University, 1992) and MS (Indian Institute of Technology, Madras, 1995) in Mechanical Engineering. His research and teaching interests are primarily in simulation-based optimization, in particular on solving large scale Markov and semi-Markov decision problems via simulation. His research has been applied to solve problems in a number of areas, such as preventive maintenance, production lot-sizing, buffer allocation in transfer lines, and airline revenue management. He has published in IIE Transactions, Management Science, Communications in Statistics, European Journal of Operational Research, and International Journal of Production Research. His research has been funded by the National Science Foundation and industry.

Tapas K. Das received his Ph.D. from Texas A&M University and now teaches at the University of South Florida where he serves as a Professor of Industrial and Management Systems Engineering. His areas of special teaching interest are applied stochastic processes, quality and reliability engineering and also service and supply systems modeling. Current research interests include simulation-based optimization of competitive and noncompetitive semi-Markov decision processes using stochastic approximation techniques, design and control of service and supply chain type systems and also the design of quality monitoring and control systems for multivariate processes using wavelet-based multiresolution analysis. He has published extensively in journals such as IIE Transactions (in all areas), IEEE Transactions on Reliability, Management Science, Communications in Statistics, Naval Research Logistics, etc. His research has been partially funded primarily by federal (NSF, in particular) and state agencies. He is a member of IIE and INFORMS.

Sudeep Sarkar received a B.Tech degree in Electrical Engineering from the Indian Institute of Technology, Kanpur, in 1988 where he was judged the best graduating Electrical Engineer. He received M.S. and Ph.D. degrees in Electrical Engineering, on a University Presidential Fellowship, from The Ohio State University, Columbus, in 1990 and 1993, respectively. Since 1993, he has been with the Computer Science and Engineering Department at the University of South Florida, Tampa, where he is currently an Associate Professor. He was the recipient of a National Science Foundation CAREER award in 1994, the USF Teaching Incentive Program Award for undergraduate teaching excellence in 1997 and the Outstanding Undergraduate Teaching Award in 1998. He is the co-author of the book *Computing Perceptual Organization in Computer Vision*, published by World Scientific. He also the co-editor of the book *Perceptual Organization for Artificial Vision Systems* published by Kluwer. He was the guest co-editor of the *Computer Vision and Image Understanding (CVIU) Journal* Special Issue on Perceptual Organization in Computer Vision, Oct 1999. He is presently serving on the Editorial Boards for the IEEE Transactions on Pattern Analysis and Machine Intelligence, and the Journal of Pattern Recognition. He served on the Editorial Board of *Pattern Analysis & Applications Journal* during 2000-2001. He has published 75 journal papers and refereed conference or workshop papers. His research interests include probabilistic reasoning, learning, Bayesian networks, and computer vision.

Contributed by the Simulation Department