Model-building Semi-Markov Adaptive Critics

Abhijit Gosavi
Department of Engineering Management & Systems Eng.
Missouri S & T
Rolla, Missouri 65401
Email: gosavia@mst.edu

Susan L. Murray
Department of Engineering Management & Systems Eng.
Missouri S & T
Rolla, Missouri 65401
Email: murray@mst.edu

Jiaqiao Hu
Department of Applied Mathematics and Statistics
Stonybrook University
Stonybrook, New York 11794-3600
Email: jqhu@ams.sunysb.edu

***Abstract*—Adaptive or actor critics are a class of reinforcement learning (RL) or approximate dynamic programming (ADP) algorithms in which one searches over stochastic policies in order to determine the optimal deterministic policy. Classically, these algorithms have been studied for Markov decision processes (MDPs) in the context of model-free updates in which transition probabilities are avoided altogether. A model-free version for the semi-MDP (SMDP) for discounted reward in which the transition time of each transition can be a random variable was proposed in Gosavi [1]. In this paper, we propose a variant in which the transition probability model is built simultaneously with the value function and action-probability functions. While our new algorithm does not require the transition probabilities *apriori*, it generates them along with the estimation of the value function and the action-probability functions required in adaptive critics. Model-building and model-based versions of algorithms have numerous advantages in contrast to their model-free counterparts. In particular, they are more stable and may require less training. However the additional steps of building the model may require increased storage in the computer's memory. In addition to enumerating potential application areas for our algorithm, we will analyze the advantages and disadvantages of model building.**

I. Introduction

Historically, the science of approximate dynamic programming (ADP) and reinforcement learning (RL) has evolved from the so-called functional equations of Bellman [2] and Howard [3], through research in "adaptive systems" [4], to ADP/RL and simulation-based optimization of MDPs [5], [6], [7], [8], [9], [10], [11]. The main goal has been to solve the Markov decision process (MDP) or some variant of the MDP for a given objective function, e.g., discounted reward, average reward, or total reward, over a finite or infinite horizon.

The so-called *model-free* algorithms have dominated the landscape of ADP/RL. These algorithms can work in simulators or in real time without the need for generating the transition probabilities of the Markov chains underlying the MDP. The main reason for this domination, of course, is that the key motivation of ADP/RL is to *avoid* the transition probabilities that lead to the curse of modeling. The curse of modeling implies that it is difficult in many real-world problems to determine the transition probabilities. However, it has been found that a class of algorithms, called *model-building* or *model-based* algorithms, in which the model is built along with the value function, can be very effective in solving certain problems, especially in the field of artificial intelligence, robotics, and recently in aviation control. The model-building algorithms can be traced to the work of Barto et al. [12] (their algorithm was dubbed RTDP, short for Real Time Dynamic Programming) and Tadepalli and Ok [13] (their algorithm was called H-Learning).

The adaptive or actor critic is one of the oldest algorithms in ADP/RL. It was first proposed in Barto et al. [14] for discounted reward MDPs in the model-free context. It predates the more popular Q-Learning algorithm [15]. The convergence of the adaptive critic was proved under some conditions in Konda and Borkar [16]. This algorithm was extended to the semi-MDP (SMDP) in [1], in which the transition time does not have to be identical for each transition, but rather can be a random variable. An assumption was also made in [1] that the immediate reward is earned as a lump sum at the start of every transition.

Hernández and Fernandez [17] also solve the problem considered in this paper via an adaptive critic algorithm. However, their algorithm exploits a uniformization approach. This approach, although well-understood in dynamic programming, is novel in the context of ADP/RL. Furthermore, their algorithm belongs to the so-called TD(λ) framework, which is more general than the one we use (TD(0)). The reader is urged to read this paper as an alternative solution model for the problem we consider.

The reader is also referred to an interesting paper by Bhatnagar and Panigrahi [18], which presented a model-free adaptive critic for hierarchical MDPs. They present a rigorous convergence analysis of their algorithm, but the problem studied in their paper has multiple decision makers (hierarchical decision making) unlike ours. Also, if their algorithm is adapted to our single decision-maker setting, it would result in a different mechanism for discounting. Unlike their model where the effective discount factor is the discount factor of MDPs (per transition) raised to a power that equals the number of state transition, we consider a time-continuous discounting process for semi-Markov control that requires the use of the exponential function (see [19] for more on this in the context of discounted semi-Markov control); further, our algorithm seeks to build the transition probability model, i.e., the expected immediate rewards and transition times, unlike theirs which is model-free.

In this paper, we combine the notions of model building

within that of an adaptive critic framework for discounted reward SMDPs. Model-building algorithms within a Q-Learning framework have been proposed in Gosavi [20], [21]. We first discuss some recent applications of model building algorithms (Section II). Thereafter, we discuss the potential advantages and disadvantages of model building in contrast to their model-free counterparts (Section III). Then, we present step-by-step details of our new algorithm (Section IV). We present some numerical results in Section V, and discuss some convergence properties in Section VI. We conclude the paper with a discussion on future work (Section VII).

II. Applications of Model Building

In this section, we provide some motivating examples of the use of model building. Many field tests of ADP/RL appear to exploit the model. Recently, model-building ADP/RL has been used in aviation control, in particular in unmanned helicopter control [22], [23]. These studies include the application of ADP/RL algorithms to train a helicopter to perform certain actions. It is known that with sufficient training pilots can prevent a helicopter from crashing by performing an emergency procedure known as autorotation. In this procedure, the pilot has to control the helicopter such that potential energy from altitude is transferred to rotor speed. Ensuring a certain speed is critical for landing safely.

Studies of the human brain via fMRI studies [24], [25] have used model-building RL algorithms rather than their model-free counterparts. The study of the human brain is an important topic in neuro-science. RL models are being increasingly used to understand how the brain functions. It turns out the model-building RL algorithms find preference in modeling these complex tasks.

A recent case study of high-speed obstacle avoidance [26] also uses model-building ADP/RL. Finally, a case study of robotic soccer in which model-building reinforcement learning is used can be found in [27]. One of the first papers in this area [13] used a model-building algorithm for training an automated guided vehicle to perform an important task of selecting jobs in an automation environment. Although the number of applications of model-building RL is significantly smaller than the corresponding number for model-free RL, it is clear that interest in model-building RL has not died and in fact appears to be increasing.

III. Advantages and Disadvantages of Model Building

Numerous tests have showed that model-building algorithms are more stable than their model-free counterparts (see e.g., [13]). In other words, they are less likely to converge to sub-optimal solutions. Model-based algorithms are also less susceptible to the need for tuning step sizes properly. And last but not the least, one reason for the failure of neural networks as function approximators with model-free algorithms like Q-Learning is the simulation noise inherent in these algorithms. Interestingly, this was highlighted some time back in [28] and [29]. Model-based algorithms, which have less noise in their main updates, can potentially overcome this difficulty when combined with neural networks.

However, model-based algorithms also come with some disadvantages. First, they require an additional step of building the model. Second, the building of the model in conjunction with function approximation can pose serious challenges. Finally, their computational burden is also higher than that of their model-free counterparts. This will be explained below.

IV. New Algorithm

We now present some background for model building. We begin with some notation. Let $\mathcal{S}$ denote the finite set of states, $\mathcal{A}(i)$ the finite set of actions permitted in state i, and $d(i)$ the action chosen in state i when policy 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 expressed as:

$$\bar{r}(i,a) = \sum_{j=1}^{|\mathcal{S}|} p(i,a,j)r(i,a,j). \quad (1)$$

The goal in Markov control is to maximize the value function for every $i \in \mathcal{S}$ over the set of all Markovian policies.

For the semi-Markov case, if one assumes that the rewards are acquired as a lump sum at the end of the transition, for the discount factor, one must use

$$\exp(-\bar{\gamma}\tau)$$

where $\bar{\gamma}$ is the rate of discounting and τ is the duration of time period over which one discounts (see [19]). Also the time spent in each state is defined as follows:

$$t(.,.,.) : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \Re \quad (2)$$

and denotes the time of one transition. The expected transition time from state i when action a is chosen in it can be expressed as:

$$\bar{t}(i,a) = \sum_{j=1}^{|\mathcal{S}|} p(i,a,j)t(i,a,j).$$

Model building in the literature has classically been based on the notion of computing counters that store the number of times a state-action pair has been tried and the number of times it has led to a given state, i.e., when one tries action a in state i and transitions to j in the simulator, one increments two counters:

$$N_a(i) \leftarrow N_a(i) + 1;$$

$$W_a(i,j) \leftarrow W_a(i,j) + 1.$$

The transition probability $p(i,a,j)$ is then estimated as:

$$\tilde{p}(i,a,j) = W_a(i,j)/N_a(i).$$

While this is intuitively appealing, both counters, $N(.)$ and $W(.)$ increase continuously to infinity as the simulation progresses, and must be stored as look-up tables, thus, ruling out combination with neural networks.

We now discuss the main updates in our new algorithm. We will suppress the superscript k in the notation for the iterates, e.g., $J(i)$, and the step sizes e.g., μ.

We will assume that the action-selection distribution $P(i,a)$ (i.e., a distribution function over the set of admissible actions of the current state i) can be parameterized in the form

$$P(i,a) = \exp(\beta(i,a))/\sum_{b\in\mathcal{A}(i)} \exp(\beta(i,b))$$

$\forall a \in \mathcal{A}(i)$. Also, our value function $J(i)$ will be updated as follows:

$$J(i) \leftarrow J(i) + \mu\times$$

$$\left[\tilde{r}(i,a) - J(i) + \exp(-\bar{\gamma}\tilde{t}(i,a))\sum_{l\in\mathcal{S}}\tilde{p}(i,a,l)J(l)\right]. \quad (3)$$

The action-selection parameter for action a in state i will be updated as follows ensuring that $\beta(i,a)$ remain bounded:

$$\beta(i,a) \leftarrow \beta(i,a) + \eta\times$$

$$\left[\tilde{r}(i,a) - J(i) + \exp(-\bar{\gamma}\tilde{t}(i,a))\sum_{l\in\mathcal{S}}\tilde{p}(i,a,l)J(l)\right]. \quad (4)$$

The above does not ensure that $\beta(i,a)$ can remain bounded. One way to work around this is to artificially bound these values as follows: Let $\beta^* > 0$ be a large number such that $\exp(\beta^*)$ is a number that the computer can handle without error. If $\beta(i,a) < -\beta^*$, set $\beta(i,a) = -\beta^*$ and if $\beta(i,a) > \beta^*$, set $\beta(i,a) = \beta^*$.

Finally, the immediate reward, the transition time, and the transition probabilities will be updated as follows:

$$\tilde{r}(i,a) \leftarrow (1-\theta)\tilde{r}(i,a) + \theta r(i,a,j) \quad (5)$$

$$\tilde{t}(i,a) \leftarrow (1-\theta)\tilde{t}(i,a) + \theta t(i,a,j); \quad (6)$$

$$\tilde{p}(i,a,j) \leftarrow \tilde{p}(i,a,j) + \theta[1-\tilde{p}(i,a,j)]. \quad (7)$$

$$\tilde{p}(i,a,l) \leftarrow \tilde{p}(i,a,l) - \theta\frac{1-\tilde{p}(i,a,j)}{|\mathcal{S}|-1};\ \text{for all } l \neq j. \quad (8)$$

In general, the step-sizes should satisfy the following rule required in multiple time scales [30]:

$$\limsup_{k\to\infty} \eta^k/\mu^k = 0; \qquad \limsup_{k\to\infty} \theta^k/\eta^k = 0, \quad (9)$$

where we use the superscript, k, to indicate that the step sizes change with every iteration. This ensures that the updating is separated on the time scales and yet it can be done simultaneously. We note that we have tested these ideas in the context of Q-Learning [21], [20], where they work efficiently.

We now present the steps in our new algorithm

A. Steps in the model-building algorithm

Step 1. Initialize the functions J, β and P. Also, initialize the transition probabilities $\tilde{p}(i,a,l) = 1/|\mathcal{S}|$ for every a. Set the number of iterations k to 1.

Step 2. Assume system is in state i. Select action a with probability $P(i,a)$. Simulate action a. Let the next state be j. Also, let the immediate reward be $r(i,a,j)$ and the transition time be $t(i,a,j)$.

Step 3. Update $J(i)$ as shown in Equation (3).

Step 4. Update $\beta(i,a)$ as shown in Equation (4) using β^* as the bound.

Step 5. Update $\tilde{r}(i,a)$, $\tilde{t}(i,a)$ and also the transition probabilities as shown in Equations (5-8). Increment k by 1.

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

Step 7. Determine the optimal policy from the value function J and stop.

B. Steps in the model-free algorithm

We also present the steps in the model-free counterpart of the above algorithm to highlight the differences.

Step 1. Initialize the functions J, β and P. Set the number of iterations k to 1.

Step 2. Assume system is in state i. Select action a with probability $P(i,a)$. Simulate action a. Let the next state be j. Also, let the immediate reward be $r(i,a,j)$ and the transition time be $t(i,a,j)$.

Step 3. Update $J(i)$ as follows:

$$J(i) \leftarrow J(i) + \mu\times$$

$$[r(i,a,j) - J(i) + \exp(-\bar{\gamma}t(i,a,j))J(j)].$$

Step 4. Update $\beta(i,a)$ as follows:

$$\beta(i,a) \leftarrow \beta(i,a) + \eta\times$$

$$[r(i,a,j) - J(i) + \exp(-\bar{\gamma}t(i,a,j))J(j)].$$

Again, $\beta(i,a)$ should be bounded as done in the model-building algorithm above.

Step 5. Increment k by 1. If $k < k_{\max}$, set $i \leftarrow j$ and return to Step 2. Otherwise go to Step 6.

Step 6. Determine the optimal policy from the value function J and stop.

A comparison of the steps in the model-building and the model-free versions shows that the model-building version requires the additional Step 5 in which the transition probabilities, the transition rewards and transition times are estimated. Clearly, these increase the computational burden. Also, the sums over the state space required in Equations (3) and (4) require a significant amount of computation. However, as stated above, the model-based updates drive the simulation noise in the updates of J and β to 0, which provides some advantages in function approximation. Also, it is not difficult to store the transition probabilities, rewards and times in neural networks or some other function approximators.

Finally, as a point of reference for classical model building, we present the RTDP algorithm for MDPs in [12].

C. Steps in RTDP

Note that the algorithm is presented for discounted reward MDPs, but can be easily adopted for SMDPs by suitable change of discount factor.

Step 1. Initialize the functions J to 0. Let $R_a(i) = 0$, $N_a(i) = 0$ for all i and $W_a(i,j) = 0$ for every a and all (i,j) pairs. Set the number of iterations, k, to 1.

Step 2. Assume system is in state i. Select action a. Simulate action a. Let the next state be j. Also, let the immediate reward be $r(i,a,j)$. Set:

$$R_a(i) \leftarrow R_a(i) + r(i,a,j).$$

Also, set

$$N_a(i) \leftarrow N_a(i) + 1$$

and

$$W_a(i,j) \leftarrow W_a(i,j) + 1.$$

Step 3. Update $\tilde{r}(i,a)$ as follows:

$$\tilde{r}(i,a) = \frac{R_a(i)}{N_a(i)}.$$

Update the transition probabilities as follows:

$$\tilde{p}(i,a,l) = W_a(i,j)/N_a(i).$$

for $l = 1, 2, \ldots, |\mathcal{S}|$.

Step 4. Update $J(i)$ as follows:

$$J(i) \leftarrow (1-\mu)J(i) + \mu\times$$

$$\left[\tilde{r}(i,a) + \lambda \sum_l \tilde{p}(i,a,l)J(l)\right].$$

Step 5. Increment k by 1. If $k < k_{\max}$, set $i \leftarrow j$ and return to Step 2. Otherwise go to Step 6.

Step 6. Determine the optimal policy from the value function J and stop.

RTDP for discounted SMDPs would require in addition the estimation of the expected time, $\tilde{t}(i,a)$, and a suitable modification of the discount factor as in Equation (3). It should be clear from a comparison of RTDP and our model-building version that we do not require the storage of counters, $N_a(i)$, $R_a(i)$ and $W_a(i,j)$. RTDP, of course, was developed for look-up tables and works well when the state space can be managed with a look-up table. Our algorithm for model building is geared toward approximation with some function approximation scheme such as regression or neural networks.

V. Numerical Results

In this section, we describe in detail the results of our experiments. Our algorithm had to be modified in order for it to overcome some numerical difficulties. We first describe this issue. Thereafter, we present the empirical results.

A. Modified algorithm

We begin with an explanation of why we modified the algorithm during our numerical experiments. During our experimentation, we discovered that obtaining the transition probabilities explicitly, i.e., as shown in Equations (7) and (8), took a very long time. Since these values are small, they are also susceptible to floating point errors. In addition, until these estimates converged to their exact values, the algorithm strayed from the optimal solution. Hence, we computed them indirectly as follows using the following term:

$$J_{next}(i,a) = \sum_{j \in \mathcal{S}} p(i,a,j)J(j),$$

for all (i,a). The above quantity was estimated, rather than estimating the transition probabilities, as follows:

$$J_{next}(i,a) \leftarrow (1-\theta)J_{next}(i,a) + \theta J(j).$$

Via the above definition of $J_{next}(.,.)$, the update in (3) and (4) can be written as:

$$J(i) \leftarrow J(i) + \mu\times$$

$$\left[\tilde{r}(i,a) - J(i) + \exp(-\bar{\gamma}\tilde{t}(i,a))J_{next}(i,a)\right] \quad (10)$$

and

$$\beta(i,a) \leftarrow \beta(i,a) + \eta\times$$

$$\left[\tilde{r}(i,a) - J(i) + \exp(-\bar{\gamma}\tilde{t}(i,a))J_{next}(i,a)\right]. \quad (11)$$

There are two merits to using the above: (1) the algorithm is less susceptible to errors in $J_{next}(i,a)$ than to errors in the transition probability estimates, and (2) unlike Equations (3) and (4) which have an inner product over the state space, Equations (10) and (11) do not.

B. Test instances and empirical results

We will use μ to denote a policy for which $\mu(i)$ will denote the (deterministic) action to be chosen in state i; e.g., $(2,1)$ will denote a policy with action 2 in state 1 and action 1 in state 2. Let $\mathbf{P}_\mu$ and $\mathbf{R}_\mu$ denote the transition probability and transition reward matrices, respectively, associated with policy μ. Also, let $\mathbf{T}_\mu$ denote the transition time matrix for policy μ. In our experiments, for the sake of simplicity, we have assumed that each transition takes a fixed amount of time, which depends on the transition. However, the algorithm should also work when the distributions of the transition times are specified.

The first test instance, which we call $smdp1$, is a 2-state SMDP with the following parameters: $\bar{\gamma} = 0.01$, and

$$\mathbf{P}_{(1,1)} = \begin{bmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{bmatrix}; \mathbf{P}_{(2,2)} = \begin{bmatrix} 0.9 & 0.1 \\ 0.2 & 0.8 \end{bmatrix};$$

$$\mathbf{R}_{(1,1)} = \begin{bmatrix} 6.0 & -5 \\ 7.0 & 12 \end{bmatrix}; \mathbf{R}_{(2,2)} = \begin{bmatrix} 10.0 & 17 \\ -14 & 13 \end{bmatrix};$$

$$\mathbf{T}_{(1,1)} = \begin{bmatrix} 1.0 & 5 \\ 120 & 60 \end{bmatrix}; \mathbf{T}_{(2,2)} = \begin{bmatrix} 50 & 75 \\ 7 & 2 \end{bmatrix}.$$

We use 3 other test instances for which the parameters are identical to those of $smdp1$ with the following exceptions: $smdp2$ — $r(1,1,2) = 25$; $smdp3$ — $p(2,2,1) = 0.3$ and $p(2,2,2) = 0.7$; $smdp4$ — $r(1,1,2) = -50$, $p(2,2,1) = 0.8$ and $p(2,2,2) = 0.2$.

The results of our numerical experiments are provided in Tables I and II. Table I presents results from the four test problems using (i) the modified adaptive critic algorithm described above and (ii) a Q-Learning algorithm for SMDPs [1]. The latter algorithm is used for comparison of the value functions. Under the step-size rules shown below, the adaptive critic did converge to the optimal solution in each case tried.

The solution obtained appears to be sensitive to the step-sizes chosen. The step-sizes should ideally follow the rules specified in (9); however, surprisingly, the rules that follow these conditions did not produce the optimal solutions in our experiments. After much experimentation, we discovered that the following rules worked:

$$\mu = \frac{log(k)}{k}; \qquad \eta = \frac{1}{100+k}; \qquad \theta = \frac{50}{100+k}.$$

Note that θ and η do not satisfy the condition in (9), mathematically required for convergence.

As stated above, using a rule for θ that would satisfy these conditions led to sub-optimal solutions in our experiments. One reason for this could be the potential difficulty identified with using more than two time scales in Borkar [31] (see pp. 67). In particular, the time scales have to be separated but at the same time need to be inter-dependent. Clearly, these rules do not obey these criteria. On the other hand, the three timescale framework permits us to solve some problems not easily solvable otherwise; see e.g., [32]. Hence, a deeper understanding of the theoretical aspects of these issues is clearly needed.

We used $\beta^* = 5.00$ and ran the adaptive critic for a maximum of 100,000 iterations. The optimal action in a given state for the adaptive critic is the action that maximizes the P-value for that state. As is clear from the results, the adaptive critic converges to the optimal solution in every case. However, the actual value function is accurately estimated only for $smdp_4$, where the P-values do not hit the artificially set thresholds, i.e., β^* or $-\beta^*$.

Table II presents values of the expected immediate reward and transition times, as estimated by our algorithm, along with the actual values (see Equations (1) and (2)). In the case of the expected immediate rewards and transition times, the values converge very close to the optimal values, showing that the model-building component of the algorithm works well.

TABLE I

THE TABLE SHOWS THE VALUE FUNCTION AND P-VALUES OBTAINED FROM OUR ALGORITHM AND THE SAME FROM A Q-LEARNING ALGORITHM [1]. HERE i DENOTES THE STATE, $J_Q(i)$ DENOTES THE VALUE FUNCTION OBTAINED FROM Q-LEARNING FOR STATE i, $\mu^*(i)$ DENOTES THE OPTIMAL ACTION IN STATE i, WHILE $P(i,a)$ AND $J(i)$ DENOTE OUTPUTS OF THE ADAPTIVE CRITIC FOR STATE i.

Case	i	$J_Q(i)$	$\mu^*(i)$	$P(i,1)$	$P(i,2)$	$J(i)$
$smdp_1$	1	95.87	1	5	0.35	119.55
$smdp_1$	2	101.83	2	-0.71	5.00	129.41
$smdp_2$	1	169.27	1	5	-0.98	227.89
$smdp_2$	2	154.59	2	-1.14	5	219.35
$smdp_3$	1	68.66	1	5	0.32	75.88
$smdp_3$	2	69.20	2	-0.19	5	77.97
$smdp_4$	1	25.59	2	-2.15	4.87	25.24
$smdp_4$	2	19.50	1	3.23	-0.77	19.08

TABLE II

THIS TABLE SHOWS SOME OUTPUTS OF THE ALGORITHM AND THEIR OPTIMAL VALUES

Case	(i,a)	$\tilde{r}(i,a)$	$\bar{r}(i,a)$	$\tilde{t}(i,a)$	$\bar{t}(i,a)$
$smdp_1$	(1,1)	2.65	2.7	2.21	2.2
$smdp_1$	(1,2)	10.78	10.7	52.78	52.5
$smdp_1$	(2,1)	9.82	10	86.11	84
$smdp_1$	(2,2)	7.57	7.6	3.00	3
$smdp_2$	(1,1)	11.77	11.7	2.21	2.2
$smdp_2$	(1,2)	11.43	10.7	55.09	52.5
$smdp_2$	(2,1)	9.82	10	86.22	84
$smdp_2$	(2,2)	7.58	7.6	3.00	3.0
$smdp_3$	(1,1)	2.67	2.7	2.21	2.2
$smdp_3$	(1,2)	10.68	10.7	52.46	52.5
$smdp_3$	(2,1)	10.28	10	80.60	84
$smdp_3$	(2,2)	4.87	4.9	3.51	3.5
$smdp_4$	(1,1)	-12.12	-10.8	2.29	2.2
$smdp_4$	(1,2)	10.71	10.7	52.54	52.5
$smdp_4$	(2,1)	10.01	10	83.87	84
$smdp_4$	(2,2)	-8.62	-8.6	6.01	6

VI. CONVERGENCE PROPERTIES

We now outline the mechanism to be used for showing convergence of the original algorithm and its modified version proposed in the previous section. The multiple time scale result in [30] will clearly form the backbone of the analysis. Also, the proof will rest in part on the existing result in [16] for the actor critic and the convergence of the semi-Markov discounted Q-Learning algorithm in [1].

The convergence arguments will be along the following lines. For *fixed* values of $\tilde{r}$, $\tilde{t}$ and $\tilde{p}$ functions, the convergence of the iterates defined in Equations (3) and (4) follows from the convergence of one of the algorithms in [16] and the analysis in [1]. The convergence of the iterates $\tilde{r}$, $\tilde{t}$ and $\tilde{p}$, which are updated in Equations (5)-(8), can be shown via standard Robbins-Monro arguments. Then, using the result in [30], all the iterates will together converge to an optimal solution.

For the convergence of the modified algorithm proposed in the previous section, some additional work will be needed because the iterate $J_{next}(i,a)$ is not independent of the iterates on the faster time scales. Showing convergence of this algorithm and determining learning rates that satisfy the conditions of multiple time scale convergence is part of future work that we intend to pursue.

VII. CONCLUSIONS

The adaptive critic is an algorithm that has had an interesting history. The algorithm studied here was introduced in [14] in 1983, but its convergence properties were not well-known until the publication of [16]. The work of Werbös [4] laid the ground work for the function approximation schemes in adaptive critics and also in general for ADP/RL with the introduction of the notion of the Bellman error [33]. Some of

the work in [4] also led to a body of literature on adaptive critics, some of which has been surveyed in [34].

This paper presents some of the preliminary work we have done in the area of model-building adaptive critics for semi-Markov control. Semi-Markov control finds applications in numerous areas of operations research, e.g., queueing problems, maintenance management (see [35],[36] and [17]), and airline revenue management (see [37] and [38]). We discussed the model-building framework in ADP/RL, and then presented a new model-building algorithm for the semi-Markov adaptive critic. We also presented the model-free adaptive critic for SMDPs and RTDP as points of reference. We discussed the advantages and disadvantages of the model-building algorithms.

Some of the future work that we intend to complete is to implement this algorithm on a large-scale problem using some function approximation scheme and compare its performance to its model-free counterpart. We will also plan to study the convergence properties of the algorithm using the two time scale framework in [30]. Finally, we intend to analyze a risk-penalized version of the algorithm which can be potentially used in safety applications and human performance modeling. Researchers modeling safety and emergency management are beginning to explore the application of human performance models (see Murray et al. [39]). We believe the use of adaptive critics, as described in this paper, will be a useful addition to algorithms for emergency response.

ACKNOWLEDGMENT

The first author would like to thank the University of Missouri Research Board for supporting this research partially.

REFERENCES

[1] A. Gosavi, "Adaptive critics for airline revenue management," in *Conference Proceedings of the Production and Operations Management Society, Dallas, TX*, 2007.

[2] R. E. Bellman, *Dynamic Programming*. Princeton University Press, Princeton, NJ, 1957.

[3] R. Howard, *Dynamic Programming and Markov Processes*. MIT Press, Cambridge, MA, 1960.

[4] P. J. Werbös, "Building and understanding adaptive systems: A statistical/numerical approach to factory automation and brain research," *IEEE Transactions on Systems, Man., and Cybernetics*, vol. 17, pp. 7–20, 1987.

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

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

[7] A. Gosavi, *Simulation-Based Optimization:Parametric Optimization Techniques and Reinforcement Learning*. Boston, MA: Kluwer Academic Publishers, 2003.

[8] J. Si, A. Barto, W. Powell, and D. Wunsch, *Learning and Approximate Dynamic Programming (Edited)*. New York, NY, USA: John Wiley and Sons, 2005.

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

[10] W. Powell, *Approximate Dynamic Programming: Solving the curses of dimensionality*. NJ, USA: Wiley-Interscience, 2007.

[11] C. Szepesvári, *Algorithms for Reinforcement Learning: Synthesis Lectures on Artificial Intelligence and Machine Learning*. Morgan Claypool Publishers, 2010.

[12] A. Barto, S. Bradtke, and S. Singh, "Learning to act using real-time dynamic programming," *Artificial Intelligence*, vol. 72, pp. 81–138, 1995.

[13] P. Tadepalli and D. Ok, "Model-based average reward reinforcement learning algorithms," *Artificial Intelligence*, vol. 100, pp. 177–224, 1998.

[14] A. Barto, R. Sutton, and C. Anderson, "Neuronlike elements that can solve difficult learning control problems," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 13, pp. 835–846, 1983.

[15] C. Watkins, "Learning from delayed rewards," Ph.D. dissertation, Kings College, Cambridge, England, May 1989.

[16] V. Konda and V. S. Borkar, "Actor-critic type learning algorithms for Markov decision processes," *SIAM Journal on Control and Optimization*, vol. 38(1), pp. 94–123, 1999.

[17] J. Ramirez-Hernández and E. Fernandez, "Control of a re-entrant line manufacturing model with a reinforcement learning approach," in *Sixth International Conference on Machine Learning*. IEEE, 2007, pp. 330–335.

[18] S. Bhatnagar and J. R. Panigrahi, "Actor-critic algorithms for hierarchical markov decision processes," *Automatica*, vol. 42, p. 637, 2006.

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

[20] A. Gosavi, "Model building for robust reinforcement learning," in *Conference Proceedings of ANNIE*. ASME Press, 2010.

[21] ——, "Reinforcement learning for model building and variance-penalized control," in *Proceedings of the Winter Simulation Conference, Austin, TX*. IEEE, 2009.

[22] A. Y. Ng, H. J. Kim, M. I. Jordan, and S. Sastry, "Autonomous helicopter flight via reinforcement learning," in *Advances in Neural Information Processing Systems 17*. MIT Press, 2004.

[23] P. Abbeel, A. Coates, T. Hunter, and A. Ng, "Autonomous autorotation of an rc helicopter,," in *International Symposium on Robotics*, 2008.

[24] W. Yoshida and S. Ishii, "Model-based reinforcement learning: A computational model and an fMRI study," *Neurocomputing*, vol. 63, pp. 253–269, 2005.

[25] S. Ishii, W. Yoshida, and J. Yoshimoto, "Control of exploitation-exploration meta-parameter in reinforcement learning," *Neural Networks*, vol. 15, pp. 665–687, 2002.

[26] J. Michels, A. Saxena, and A. Ng, "High speed obstacle avoidance using monocular vision and reinforcement learning," in *Proceedings of the 22nd International Conference on Machine Learning, Bonn, Germany*, 2005.

[27] M. A. Wiering, R. P. Salustowicz, and J. Schmidhuber, "Model-based reinforcement learning for evolving soccer strategies," in *COMPUTATIONAL INTELLIGENCE IN GAMES*. Springer Verlag, 2001.

[28] P. Werbös, "A menu of designs for reinforcement learning over time," in *Neural Networks for Control*. MIT Press, MA, 1990, pp. 67–95.

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

[30] V. S. Borkar, "Stochastic approximation with two-time scales," *Systems and Control Letters*, vol. 29, pp. 291–294, 1997.

[31] ——, *Stochastic approximation: A dynamical systems viewpoint*. New Delhi, India: Hindusthan Book Agency, 2008.

[32] K. Kulkarni, A. Gosavi, S. Murray, and K. Grantham, "Semi-Markov adpative critic heuristics with application to airline revenue management," 2010, accepted to the *International Journal of Control and Applications*.

[33] P. Werbös, "Consistency of HDP applied to a simple reinforcement learning problem," *Neural Networks*, vol. 3, pp. 179–189, 1990.

[34] S. Ferrari and R. Stengel, "Model-based adaptive critic designs," in *Learning and Approximate Dynamic Programming (edited by J. Si, A. Barto, W. Powell, and D. Wunsch, Chapter 3)*. New York, NY, USA: John Wiley and Sons, 2005.

[35] T. Das, A. Gosavi, S. Mahadevan, and N. Marchalleck, "Solving semi-Markov decision problems using average reward reinforcement learning," *Management Science*, vol. 45(4), pp. 560–574, 1999.

[36] A. Gosavi, "Reinforcement Learning for long-run average cost," *European Journal of Operational Research*, vol. 155, pp. 654–674, 2004.

[37] A. Gosavi, N. Bandla, and T. K. Das, "A reinforcement learning approach to a single leg airline revenue management problem with multiple fare classes and overbooking," *IIE Transactions*, vol. 34, pp. 729–752, 2002.

[38] A. Gosavi, "A reinforcement learning algorithm based on policy iteration for average reward: Empirical results with yield management and convergence analysis," *Machine Learning*, vol. 55(1), pp. 5–29, 2004.

[39] S. Murray, K. Ghosh, and M. Gosakan, "Human performance modeling for emergency management decision making," 2010, to appear in *Journal of Emergency Management*.