JAISCR, 2012, Vol. 2, No. 1, pp. 43– 58

MODEL-BUILDING ADAPTIVE CRITICS FOR SEMI-MARKOV CONTROL

Abhijit Gosavi,[1] Susan Murray,[2] Jiaqiao Hu[2] and Shuva Ghosh[1]

[1] *Department of Engineering Management and Systems Engineering*
Missouri University of Science and Technology, Rolla, MO 65401
e-mail: gosavia@mst.edu

[2]*Department of Applied Mathematics and Statistics*
Stonybrook University, Stonybrook, New York 11794-3600

Abstract

Adaptive (or actor) critics are a class of reinforcement learning algorithms. Generally, in adaptive critics, one starts with randomized policies and gradually updates the probability of selecting actions until a deterministic policy is obtained. Classically, these algorithms have been studied for Markov decision processes under model-free updates. Algorithms that build the model are often more stable and require less training in comparison to their model-free counterparts. We propose a new model-building adaptive critic, which builds the model during the learning, for a discounted-reward semi-Markov decision process under some assumptions on the structure of the process. We illustrate the use of our algorithm with numerical results on a system with 10 states and a real-world case-study from management science.

1 Introduction

The science of reinforcement learning (RL) or approximate dynamic programming (ADP) has evolved from the Bellman equation [4] and the Poisson equation [25], through research in adaptive systems [47], to ADP/RL and simulation-based optimization of MDPs [6, 41, 18, 39, 12, 35, 43].

The underlying goal in these problems is generally to solve the Markov decision process (MDP) or some variant of it for a given objective function, e.g., discounted reward, average reward, or total reward, over a finite or infinite horizon.

A subset of these problems consider the scenario that arises in discrete-event systems, where the time between two successive events (when the system state changes) is a non-zero, discrete variable. Examples of this scenario include numerous problems arising in robotics, many management science problems, and problems in queueing/communication networks. In this paper, we are interested in problems arising in discrete-event stochastic systems.

The work of Bellman and Howard has given birth to the field of dynamic programming (DP), which seeks to obtain the optimal value function in solving the MDP. DP requires the values of the transition probability matrices underlying the MDP. The discipline of ADP/RL has emerged more recently. Its landscape has been dominated by *model-free* algorithms that seek to avoid the transition probability matrices while updating the value function. A motivation for avoiding them is that they are huge for large-scale problems, making their storage and processing cumbersome. Further, ADP/RL avoids these matrices via the stochastic approximation algorithm that it employs for updating the value function.

Scattered in the ADP/RL literature one finds algorithms that build the transition probability model, either simultaneously with the stochastic-approximation-based update of the value function,

or separately first building the model using sample experience and subsequently using DP in an offline manner. The former class of algorithms are collectively referred to as *model-building* algorithms while those belonging to the latter class are known as *model-based* algorithms. These algorithms have some noteworthy advantages over their model-free counterparts [49], the principle ones being: (i) they are more stable and (ii) they require less training (sample experience). Of course, in contrast to model-free algorithms, they have an additional step of building a model, which involves certain complications that we will discuss below. But some recent studies in engineering (cited below) point towards the fact that exploiting the model may actually be beneficial in some applications. Further, a review of the literature indicates that interest in model-based/model-building algorithms has been alive for a long time in ADP/RL, despite the fact that they involve an additional layer of building the model, and continues to grow; see e.g., [45].

Contributions of this paper: Two major complications arise in building the model directly and then using it within the Bellman equation update [2, 44]. First, estimating the transition probabilities via maximum likelihood estimation may become cumbersome for large problems. Second, when the transition probability model is used within the value function update, it requires an inner product summation over the state space, which slows down every update due to the intensity of the calculations required in a large state space. In this paper, we develop a model-building actor critic that (i) avoids the cumbersome maximum likelihood estimation via storage of numerous counters and (ii) does not require the computationally intensive inner product summation over the entire state space. This is the first contribution. The second contribution lies in that our algorithm is designed for a specific type of semi-Markov decision process (SMDP) which a more general version of the MDP; i.e., our algorithm can be used not only for MDPs, but also in addition for a more general version of the MDP. Finally, our third contribution is to study the computational aspects of our algorithm in a real-world case study from management science.

The rest of this article is organized as follows. Section 2 provides a review of the relevant literature. Section 3 discusses the maximum likelihood estimation of classical model-building. Section 4 presents details of our algorithm. Numerical results are presented in Section 5. Conclusions and directions for future research are discussed in Section 6.

2 A Literature Review

In this section, we review literature on model building and actor critics that is relevant to our work.

2.1 Model-building/based RL

The papers of [2] (Real Time Dynamic Programming) and [44] (H-Learning) can be regarded as the pioneering works in model-building RL for discrete-event systems. The main idea in these algorithms is to use maximum likelihood estimation via counters. These algorithms are based on estimating the value function rather than the Q-function of RL. Model-building Q-Learning algorithms have been suggested in [21, 22, 18]. For systems where the state can change continuously, i.e., the system is not of a discrete-event type, and the state/action space are continuous, a nice account has been provided in [17].

A subset of the literature [42, 33, 27, 10, 40, 16, 45], which is model-based rather than model-building, seeks to approximate the model from sample experience (either in the real world or within a simulator) and then use DP in an offline manner. Representing the model and building it via maximum likelihood estimation using counters is usually difficult for large-scale problems. One may generate approximate models on a reduced state-action space that may be easier to handle, but their performance is usually limited by the scale of the approximation in the model [27].

And yet, many recent field tests of ADP/RL appear to exploit the model. Some of these tests are of the model-based type. A case study of robotic soccer in which model-based RL is used can be found in Wiering et al. [49]. A more recent example is from the study of controlling an unmanned helicopter [34, 1, 29]. It is known that with sufficient training, pilots can prevent a helicopter from crashing by performing an emergency procedure called auto-rotation. The goal of these ADP/RL studies was to develop an algorithm that can train a he-

licopter to perform an auto-rotation, among other maneuvers, on its own. The study of the human brain is an important topic in neuro-science, and RL models are being increasingly used to understand how the brain functions. Recently, function magnetic imaging resonance (fMRI) studies in [50, 26] have used model-based RL algorithms rather than their model-free counterparts. Finally, a case study of high-speed obstacle avoidance [32] also uses model-based algorithms.

2.2 Adaptive critics

The model-free adaptive or actor critic is one of the oldest algorithms in ADP/RL. It was first proposed in Barto et al. [3] for discounted reward MDPs. It predates the more popular Q-Learning algorithm [46]. The convergence of the model-free adaptive critic was proved under some conditions in [28]. This algorithm was extended to SMDPs in [30, 20]. Hernández and Fernandez [36] also solve the problem considered in this paper via a model-free adaptive critic algorithm that exploits a discretization approach and uses the TD(λ) framework. Finally, an interesting model-free adaptive critic for hierarchical MDPs has been proposed in [7] where the mechanism of discounting is unlike ours.

3 Maximum Likelihood Estimation

In this section, we present a discussion of the maximum likelihood estimation of the transition probabilities underlying the model. The motivation for this discussion is to expose the difficulties posed by this estimation in RL. We will introduce some notation first.

3.1 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. A stationary deterministic policy x is an $|\mathcal{S}|$-tuple that will select action $x(i)$ in state i.

3.2 Counters

One approach to maximum likelihood estimation is to use counters and then use direct estimation of means via averaging. The main idea for estimating the mean expected rewards and the transition probabilities with counters is as follows. One initializes the two functions $R_a(i)$ and $N_a(i)$ for all $i\in\mathcal{S}$ and the function $W_a(i,j)$ to 0 for every $a\in\mathcal{A}(i)$ and all (i,j) pairs. Then the system is either simulated or observed in the real world. When the system is in state i, if action a is selected and the next state is j, we perform the following update:

$$N_a(i)\leftarrow N_a(i)+1 \text{ and } W_a(i,j)\leftarrow W_a(i,j)+1.$$

Further, the transition probabilities are updated as follows:

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

for $l=1,2,\ldots,|\mathcal{S}|$, where $\tilde{p}(.,.,.)$ denotes the maximum likelihood estimate of the transition probability. This step requires the storage of counters, $N_a(.)$ and $W_a(,.,)$, in addition to the updates defined above. Classical model-building algorithms [2, 44] have used the above approach for building the model. In order to update the value function, one must also estimate the immediate reward model, which can be done as follows. Set $R_a(i)=0$ for all $i\in\mathcal{S}$ at the start. When the transition from i to j under the influence of a occurs, perform the following updates:

$$R_a(i)\leftarrow R_a(i)+r(i,a,j) \text{ and } \tilde{r}(i,a)=\frac{R_a(i)}{N_a(i)},$$

where $\tilde{r}(i,a)$ denotes the maximum likelihood estimate of the immediate reward earned in state i when action a is selected and $r(i,a,j)$ denotes the sample immediate reward earned when action a is selected in state i and the system transitions to j.

The above estimates of the transition probability and the immediate reward allow us to use the following update for the value function, $J(i)$, of an MDP:

$$J(i)\leftarrow(1-\eta)J(i)+\eta\times$$
$$\max_{a\in\mathcal{A}(i)}\left[\tilde{r}(i,a)+\lambda\sum_{l\in\mathcal{S}}\tilde{p}(i,a,l)J(l)\right], \qquad (1)$$

where η is the learning rate or the step size. A drawback of the above is that it requires an inner product summation over $\mathcal{S}$ for every action, which is likely to be time consuming for large-scale problems, in addition to a maximization over the action space $\mathcal{A}(i)$.

3.3 Robbins-Monro stochastic approximation

The Robbins-Monro algorithm [37] allows one to bypass the storage of counters. For estimating the immediate rewards, we can use the following scheme based on the Robbins-Monro algorithm:

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

where θ is a learning rate that is gradually decayed to 0. Clearly, scheme of this kind does not need counters such as $R_a(i)$ and $N_a(i)$. In the new algorithm that we propose in the next section, we will use the above mechanism to estimate the expected immediate reward and thereby avoid the counters $R_a(.)$, and $N_a(.)$. We will also avoid storing the transition probabilities directly thereby bypassing the counter $W_a(.,.,.)$; furthermore, not storing the transition probabilities directly will also allow us to perform the Bellman update without the inner production summation and maximization over the action space in Equation (1). Counters increase the storage burden of the algorithm, while the inner product summation makes every iteration of the algorithm computationally intensive.

4 New Algorithm

This section presents some of the background needed for our new algorithm and its detailed description. In subsection 4.1, we present a quick overview of the SMDP that we study and its underlying Bellman equation. Thereafter, in subsection 4.2, we present an adaptation of the classical model-free adaptive critic for our SMDP. Finally in subsection 4.3, we present a step-by-step description of our new algorithm.

4.1 SMDP

In an MDP, the time spent in each transition is assumed to be 1 since it is the same for every transition. In an SMDP, however, the time is a part of the model, and cannot be assumed to be 1 for every transition. For instance in queueing problems, in systems prone to failures, and in many wireless communication problems, the transition times are rarely the same. In case of discounted rewards, the transition time also affects the value of the discount factor and hence must be incorporate into the model. We study the SMDP under two assumptions that we state next.

Assumption 1 *The transition reward is earned in a lump sum at the start of the transition.*

Assumption 2 *The time spent in each transition is a deterministic variable.*

Assumption 1 avoids continuously awarded rewards by forcing the reward to be awarded only once, while Assumption 2 provides that the time of transition is not necessarily one but of fixed duration. Note, however, that Assumption 2 does not imply that the time of transition is the same for every transition. A more general model for the SMDP would assume that the reward could be earned continuously during the transition and that the time of transition could be a random variable. Here, in order to keep our development tractable, we make these simplifying assumptions. Both assumptions hold for any MDP.

We now present some additional notation that we will need. The *expected* immediate reward earned in state i when action a is chosen in it can be expressed as:

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

Also the time spent in each state is defined as follows:

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

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:

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

For the SMDP, the discount factor is defined as $e^{-\gamma\tau}$, where γ is the rate of discounting and τ is the duration of time period over which one discounts (see [5]). The goal in the SMDP here is to maximize the

total discounted reward over an infinite time horizon. To this end, one solves the Bellman equation provided in the next result that follows directly from Prop. 1.2.2 from Bertsekas [5, vol 2.].

Theorem 1 *Under Assumptions 4.1 and 4.1, for all $i \in \mathcal{S}$, the optimal value function J^* for the SMDP satisfies: For all $i \in \mathcal{S}$*

$$J^*(i) = \max_{a \in \mathcal{A}(i)} \left[\bar{r}(i,a) + \sum_{j \in \mathcal{S}} p(i,a,j) e^{-\gamma t(i,a,j)} J^*(j) \right]. \quad (2)$$

Furthermore, J^* is the unique solution of this equation.

Our algorithm must hence be designed to solve the Bellman equation in (2).

4.2 A model-free version

In order to motivate our discussion of a model-based adaptive critic for the SMDP, we present an overview of the model-free adaptive critic for MDPs and its model-free adaptation for the SMDP at hand. The model-free version will then be modified to obtain a model-based algorithm for the SMDP.

In an adaptive critic, one starts with a randomized policy in which the action a is selected in state i using probability $P(i,a)$. $P(i,a)$ is defined using the action-selection parameter $\beta(i,a)$as follows:

$$P(i,a) = \frac{e^{\beta(i,a)}}{\sum_{b \in \mathcal{A}(i)} e^{\beta(i,b)}} \quad \forall a \in \mathcal{A}(i). \quad (3)$$

The algorithm updates $\beta(i,a)$ in a manner such that asymptotically one reaches an optimal deterministic policy. The value function of the randomized policy, $J(.)$, is also updated along with $\beta(,.,)$. The main updates from the two timescale algorithm in [28] is as follows. Actions are selected randomly using the function $P(,.,)$. When the transition from i to j under action a occurs, perform the following updates:

$$\beta(i,a) \leftarrow \beta(i,a) + \mu \times$$
$$\left[r(i,a,j) - J(i) + e^{-\gamma t(i,a,j)} J(j) \right]; \quad (4)$$
$$J(i) \leftarrow J(i) + \eta \times$$
$$\left[r(i,a,j) - J(i) + e^{-\gamma t(i,a,j)} J(j) \right]. \quad (5)$$

In Equation (4), η is a learning rate different than μ; in fact, η should decay to 0 faster than μ. Also, in Equation (4), $\beta(,.,)$ has to be artificially constrained within $[-\beta^*, \beta^*]$, since it can become unbounded otherwise, where $\beta^* > 0$ is a scalar small enough such that e^{β^*} can be stored in the computer without overflow. The updating of $\beta(,.,)$ will thus work as follows. If $\beta(i,a) < -\beta^*$, set $\beta(i,a) = -\beta^*$; if $\beta(i,a) > \beta^*$, set $\beta(i,a) = \beta^*$; otherwise, use Equation (4) for updating $\beta(i,a)$.

4.3 Steps in the new algorithm

In order to develop a model-based version from the model-free version presented above, we need to define the function J' as follows: $J'(.,.) : \mathcal{S} \times \mathcal{A} \rightarrow \Re$ where

$$J'(i,a) = \sum_{j \in \mathcal{S}} p(i,a,j) J(j). \quad (6)$$

Thus $J'(i,a)$ can be viewed as the expected value of $J(j)$, the value function of the next state j, when action a is selected. Our goal in the model-building adaptation of the model-free version is to replace the samples by estimates of the expected values. Hence, $r(i,a,j)$ will be replaced by $\tilde{r}(i,a)$ and $J(j)$ by $J'(i,a)$ when action a is chosen. Further, we need to replace the sample $t(i,a,j)$ by $\tilde{t}(i,a)$. We thus obtain the following updates from transforming those in (4) and (5):

$$\beta(i,a) \leftarrow \beta(i,a) + \mu \times$$
$$\left[\tilde{r}(i,a) - J(i) + e^{-\gamma \tilde{t}(i,a)} J'(i,a) \right]. \quad (7)$$
$$J(i) \leftarrow J(i) + \eta \times$$
$$\left[\tilde{r}(i,a) - J(i) + e^{-\gamma \tilde{t}(i,a)} J'(i,a) \right]. \quad (8)$$

We note that $\beta(,.,)$ will have to be bounded as in the model-free case. Of course, we must in addition develop a mechanism to estimate the functions $\tilde{r}(,.,)$, $\tilde{t}(.,.)$ and $J'(,.,)$. These functions will be estimated in the standard averaging process used in a stochastic approximation algorithm. We will present the details related to these functions in the step-by-step description that follows.

We will use three different learning rates, η, μ, and θ in the algorithm below. The learning rates will change with the iteration number, k, as is standard in RL. We can denote the learning rate with the enhanced notation, e.g., μ_k. However, we drop the subscripts in our description to increase clarity. The

learning rates must obey the following standard assumptions needed for stochastic approximation [6]:

$$\sum_k \eta_k = \infty; \sum_k \mu_k = \infty; \sum_k \theta_k = \infty;$$

$$\sum_k (\eta_k)^2 < \infty; \sum_k (\mu_k)^2 < \infty; \sum_k (\theta_k)^2 < \infty.$$

In other words, the learning rates should sum up to ∞, while their squares should sum to a finite value.

Step 1. Initialize $J'(i,a) = 0$, $\beta(i,a) = 0$, $\tilde{r}(i,a) = 0$ and $\tilde{t}(i,a) = 0$ for all (i,a) pairs and initialize $J(i) = 0$ for every $i \in \mathcal{S}$. Set the number of iterations k to 1. Set $k_{\max}$ to a suitable number. Initialize β^* to a largest positive value such that can be e^{β^*} can be stored in the computer without overflow.

Step 2. Assume the system to be in state i. Select action a with probability $P(i,a)$, where $P(i,a)$ is computed using Equation (3). 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 $\beta(i,a)$ as shown in Equation (7) using using β^* as the bound as discussed in Section 4.2.

Step 4. Update $J(i)$ as shown in Equation (8).

Step 5. Update $\tilde{r}(i,a)$, $\tilde{t}(i,a)$ and $J'(i,a)$ as shown below:

$$\tilde{r}(i,a) \leftarrow \tilde{r}(i,a) + \theta[r(i,a,j) - \tilde{r}(i,a)]; \quad (9)$$

$$\tilde{t}(i,a) \leftarrow \tilde{t}(i,a) + \theta[t(i,a,j) - \tilde{t}(i,a)]. \quad (10)$$

$$J'(i,a) \leftarrow J'(i,a) + \theta[J(j) - J'(i,a)]. \quad (11)$$

Increment k by 1.

Step 6. If $k < k_{\max}$, set $i \leftarrow j$, update η, μ, and θ, and return to Step 2. Otherwise, if $k = k_{\max}$, stop learning and go to Step 7.

Step 7. Determine an optimal action, $d(i)$, for state i for every $i \in \mathcal{S}$ as follows:

$$d(i) \in_{a \in \mathcal{A}(i)} P(i,a).$$

Then d will denote an optimal policy.

Note that our algorithm not only avoids the inner product summation over $\mathcal{S}$ required in the maximum likelihood estimation but also avoids counters such as $R_a(.)$, $T_a(.)$, and $N_a(.)$. The immediate reward and the immediate transition time are estimated using the standard Robbins-Monro algorithm of stochastic approximation via updates in Equations (9) and (10) respectively. We further note that by setting $\tilde{t}(i,a)$ to 1, we can use the above algorithm for any MDP.

5 Numerical Results

In this section, we present results of the use of our algorithm on some small-scale problems with ten states and two actions (subsection 5.1) and a real-world case study from management science related to total productive maintenance (subsection 5.2).

5.1 Small-scale problems

In this subsection, we study the behavior of our new algorithm on a small-scale problem with 10 states and 2 actions in each state. Our goal is to compare the performance of our new algorithm with that of Q-Learning in terms of the quality of the solution (policy) generated, and also the value function and elements of the model (immediate reward and transition time).

We now define some notation needed to present the inputs for the 10-state SMDPs. We will use $\mathbf{P}_z$, $\mathbf{R}_z$ and $\mathbf{T}_z$ to denote the transition probability matrix, the transition reward matrix, and the transition time matrix respectively for a stationary, deterministic policy z. We now define two policies, x and y, as follows:

$$x = (1,1,1,1,1,1,1,1,1,1);$$

$$y = (2,2,2,2,2,2,2,2,2,2).$$

The matrices, $\mathbf{P}, \mathbf{R}$ and $\mathbf{T}$ associated with x and y, are defined in the Appendix. They will be the input parameters for case 1, the first 10-state SMDP that we studied. The input parameters for the other nine cases are defined in Table 7. We used $\gamma = 0.1$ for all the cases and $\beta^* = 15$. The learning rates that we used are:

$$\mu = \frac{1}{100+5k}; \eta = \frac{\log(k+1)}{k+1}; \theta = \frac{50}{100+k}.$$

In order to numerically show that our algorithm finds the optimal policy and to benchmark its performance against an existing algorithm, we also ran Q-Learning for the SMDP under these assumptions [9, 23]. The main transformation in the Q-Learning algorithm for the SMDP we study would be as follows:

$$Q(i,a) \leftarrow Q(i,a) + \mu \times$$

$$\left[r(i,a,j) - Q(i,a) + e^{-\gamma \bar{t}(i,a,j)} \max_{b \in \mathcal{A}(j)} Q(j,b) \right].$$

Then, the value function obtained via Q-Learning can be defined as follows: $J_Q(i) \equiv \max_{a \in \mathcal{A}(i)} Q(i,a)$. Table 1 shows the value function from our new algorithm and that from Q-Learning for Case 1. As is clear, our algorithm produces values very close to those from Q-Learning, which is an encouraging finding. The relative performance for all the other 9 cases is listed in Table 2, which shows that in all these cases also our algorithm performs extremely well. Table 3 compared the values of the expected immediate rewards and expected transition times estimated from our new algorithm and their actual values for Case 1. The optimal policies for all the cases are presented in Table 4; our algorithm produced the optimal policy in every case.

Table 1. Value function from new algorithm and Q-Learning for case 1

i	$J(i)$	$J_Q(i)$
1	8.001519	8.572999
2	13.515785	13.948835
3	12.146564	13.416791
4	16.337069	16.080918
5	13.460989	15.248618
6	20.349903	20.578502
7	21.953309	22.228289
8	11.055585	13.075585
9	8.395198	9.337231
10	-1.709660	-0.626395

Table 3. Results from all cases

Case	$\|\|J(i) - J_Q(i)\|\|_\infty$
1	2.020000
2	1.800999
3	2.066055
4	0.970520
5	1.812913
6	2.107958
7	2.300655
8	1.822307
9	1.883836
10	1.813317

Table 4. Optimal policies for all cases.

Case	Optimal policy
1	$(2,1,2,1,1,2,2,2,2,2)$
2	$(2,1,2,1,1,2,2,2,2,2)$
3	$(2,1,2,1,1,2,2,2,2,2)$
4	$(2,1,2,1,1,2,1,2,2,1)$
5	$(2,1,2,1,1,2,1,2,2,1)$
6	$(2,2,2,1,1,2,2,2,2,2)$
7	$(2,1,1,1,1,2,2,2,2,2)$
8	$(2,1,2,1,2,2,2,2,2,2)$
9	$(2,1,2,1,1,2,2,2,2,2)$
10	$(2,1,2,1,1,2,2,2,2,1)$

5.2 Management case study

An important program within production management is called Total Productive Maintenance (TPM) (see [31] for a review). The overall goal of this program is to preventively overhaul machines, especially those that are critical for ensuring that production rates meet desired targets, so as to minimize the probability of unexpected machine failures that can disrupt production. Typically, the probability of failure of a machine increases as it ages. When a failure occurs, it usually occurs without warning and almost invariably upsets production schedules and the ability of the production manager to meet deadlines if the machine is a bottleneck or crucial for production. One option to reduce the frequency of such failures is to preventively maintain the machine when it is functioning perfectly.

Table 2. Values of expected immediate reward and transition times for case 1. Here d denotes the optimal policy.

i	$d(i)$	$\bar{t}(i,d(i))$	$\tilde{t}(i,d(i))$	$\bar{r}(i,d(i))$	$\tilde{r}(i,d(i))$
1	2	42.84	42.04	6.88	6.82
2	1	32.64	33.92	9.64	9.58
3	2	27.53	27.42	9.27	9.17
4	1	27.06	26.64	11.19	11.13
5	1	16.33	16.22	9.97	9.89
6	2	34.92	34.65	18.42	18.53
7	2	39.74	39.29	19.91	20.39
8	2	24.04	24.27	9.60	8.89
9	2	40.91	41.15	7.46	7.68
10	2	38.54	38.83	−2.57	−2.37

In other words, a machine that is up and running is shut down for a pre-specified time period, checked for potential problems, such as lubrication or aging bearings, and then allowed to get back into the production mode. In general, the cost for maintenance is much lower than that for repair, which can be attributed to the fact that repairs lead to loss of production in addition to the fact that they require additional time needed for troubleshooting the problem.

It has been known in the industry that a preventive maintenance can significantly reduce the frequency of repair-induced shutdowns, thereby reducing variability in throughput time and inventory in the system. However, determining the exact time interval after which maintenance should be performed has been a challenging problem for many years. The difficulties in solving this problem arises from the fact that the underlying failure mechanisms tend to be different for every system.

The problem of determining the optimal time of maintenance are not limited to production systems. Similar issues are encountered in operation of electrical power systems (see e.g., [11, 13]) and critical infrastructure such as roads and bridges (see e.g., [38]). Interestingly, the problems underlying maintenance optimization have been frequently modeled as MDPs.

In this paper, we study a production line from an industry in upstate New York where a repair or maintenance activity typically shut down an entire production line. Repairs were significantly costlier than maintenance and also took longer since they were caused by unscheduled breakdowns. The production system consisted of numerous lines, but each line had its own unique characteristics. When it was determined that a line would be maintained, all the machines in it were subjected to maintenance. Similarly, when a machine failed in a line, generally, the entire line was affected. Managers were interested in minimizing the frequency of repairs in the production lines in order to increase productivity. The costs for repair and maintenance, as well the time it took to perform the maintenance and repair activities, were estimated. The time between successive failures was also recorded in order to determine the distribution of the time between failures. We note that we have altered some aspects of this data set in order to protect the confidentiality of the manufacturer.

We make some assumptions that are typical of maintenance modeling. We assume that the state of the system is defined by the number of production cycles since the last repair or maintenance. With such a definition of the state, it is possible to show that the state dynamics follow a Markov chain. We assume that the time between failures follows the same distribution after every repair/maintenance, i.e, the line is as good as new when it is repaired or maintained. We also assume that the cost of repair and maintenance are known with certainty. The duration of each production cycle in the data we observed had little variability and will was assumed to be deterministic. The time for repair and the time for maintenance was also approximately a multiple (not necessarily integer) of the time needed for one

production cycle. We now introduce some notation that we will need in order to present our case study:

C_r: Cost of one repair

C_m: Cost of one maintenance activity

t_p: Duration of one production cycle

M_r: Coefficient for repair time

M_m: Coefficient for maintenance time

(n,λ): Parameters of an Erlang distribution used for the time between successive failures

Some comments in regards to the notation above are in order. We assume that when a line is not in use it does not age. When it is maintained or repaired, its age is set to zero. The time between successive failures is the age at which the machine fails. It is typically a random failure with increasing failure rates. We assumed the failure time data to have an Erlang distribution, but note that our data has been modified to protect the interests of the source. We also note that our algorithm should work for any distribution since it is simulation-based. For the Erlang distribution, the mean is n/λ and the variance is n/λ^2. Also, the notation for the repair and maintenance coefficients is to be interpreted as follows. A repair is assumed to end $M_r t_p$ time units after the production cycle (during which failure occurs) starts, while a maintenance is assumed to end $M_m t_p$ time units after the maintenance activity is initiated.

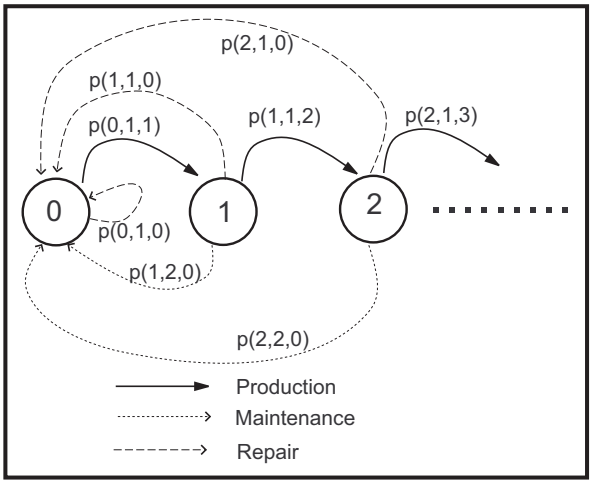

Figure 1. The Markov chain underlying the maintenance case study. Note that $p(i,a,j)$ denotes the transition probability from i to j under a, and $a = 1$ denotes production, while $a = 2$ denotes maintenance.

Transition probability structure: The Markov chain embedded in the semi-Markov process underlying the problem has been depicted in Figure 1. We now explain the transition probabilities and the (transition) reward and time structure. The state of the system is defined by the number of production cycles since last repair or maintenance. The number inside a circle (the state in a Markov chain) denotes the state. There are two actions, Produce and Maintain, to choose from in each state. When the action to produce is selected in state i, one either goes to $(i+1)$ if the production is successfully completed or returns to state 0 if the machine fails before the production is complete. The transition probability $p(i,1,0)$ depends on the distribution of the time between failures and the state i. The transition in this case involves the time spent in the partial production and the repair of the machine. When the action to maintain is chosen in a state, one returns to the state 0 with probability 1. In this case, the transition involves the time spent in maintaining the machine. Using the notation above, we have that $t(i,1,i+1) = t_p$, $t(i,1,0) = M_r t_p$, and $t(i,2,0) = M_m t_p$.

Table 5 presents input data for some of the parameters needed for 10 different systems that we studied. The units of t_p and λ are *hours* and $hour^{-1}$ respectively, while those of C_m and C_r are dollars. The different cases were created using industrial data, some of which has been modified [19]. Note that the discount factor $e^{\gamma} \approx 1/(1+R)$, where R is the rate of interest per unit time.

Table 6. The optimal policy is the age in hours of the production line when maintenance is recommended.

Case	Optimal Policy (hours)
1	60
2	70
3	72
4	120
5	60
6	70
7	80
8	40
9	50
10	70

Table 5. Input parameters: $\lambda = 0.1$ and $\beta^* = 10$ in all our experiments.

Case	C_m	C_r	γ	t_p	M_m	M_r	n
1	2	10	0.1	10	3	1.25	8
2	2	10	4×10^{-5}	10	3	1.25	8
3	2	10	0.1	12	3	1.25	8
4	2	10	0.1	15	3	1.25	8
5	2	10	0.1	10	2.25	1.25	8
6	2	10	0.1	10	3	1.25	6
7	2	10	0.1	10	3	1.25	7
8	2	10	0.1	20	3	1.25	8
9	2	15	0.1	5	3	1.25	8
10	2	15	0.1	10	3	0.25	8

For case 2, we used an interest rate of 12 percent per annum, assuming we have 3000 operational hours in one year. For the other 9 cases, we used a value often used in the literature. For all our experiments, we used learning rates defined in the previous subsection. Table 6 shows the policy determined by our new algorithm in terms of the age at which it recommends maintenance. The policy matches that produced by Q-Learning. Each case was run for 10^6 hours of simulation time for learning, and it took less than 1 millisecond on an Intel Pentium Processor with a speed of 2.66 GHz on a Red-Hat Linux operating system. Table 6 shows that the optimal policy for preventive maintenance varies from stopping the production every 40 hour for PM to stopping it every 120. This range shows the variability in production systems and the importance of modeling such systems. If management chooses a heuristic PM policy such as performing maintenance every 70 hours based on say the overall average rates, significant costs would be incurred. Cases such as 1 and 2 will then incur extra costs from system breakdowns since the typical time between failure is less the 80 hours chosen by management. On the other hand, for the case with a higher time between failures (such as case 3 and 4) the company will waste money maintaining the production system more frequently than needed.

6 Conclusions

The problem of developing model-building RL algorithms is a long-standing one. It has been known for some time that the noise in the model-free update can cause function approximation to become unstable with neural networks [48]. Recently, remarkable success has been observed on the helicopter domain using model-based approaches that seek to build the model from sample experience and thereafter use DP offline. However, model-building algorithms have lagged behind primarily due to the fact that storing the transition probabilities using neural networks becomes cumbersome for large-scale problems and performing an inner product summation over the entire state space can slow down the updating process. In this paper, we seek to develop a model-building algorithm that avoids the transition probabilities while using the expectation rather than the sample within the main Bellman update, thereby avoiding the computationally intensive inner product summation. Also, our algorithm is amenable to function approximation, via for instance neural networks, since it does not store the transition probabilities. To the best of our knowledge, this is also the first algorithm that seeks to use the adaptive critic in a model-building exercise. Furthermore, we were able to extend the applicability of the algorithm from the MDP for which adaptive critics have been studied for the most part to an SMDP in which the transition time is deterministic and the reward is earned as a lump sum at the start of the transition.

We obtained very encouraging numerical results with our algorithm. It generated optimal solutions in the 10-state SMDPs that we studied and produced value functions remarkable close to those of Q-Learning. We were also able to test the algorithm on a real-world problem of maintenance optimization based on data gathered from an automobile firm in New York state.

The algorithm statement without the supporting theory or the numerical results has appeared in the conference proceedings of the IEEE symposium series on computation intelligence (ADPRL) [24]. Future directions for this research will include

1 developing a convergence proof for the algorithm using the multiple time scale framework of Borkar [8]

2 extending the problem to production-inventory maintenance problems of the nature studied in [14, 15]

3 an extension of the algorithm to a more general model of the SMDP which does not need Assumptions 1 and 2

References

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

[2] A.G. Barto, S.J. Bradtke, and S. Singh. Learning to act using real-time dynamic programming. *Artificial Intelligence*, 72:81–138, 1995.

[3] A.G. Barto, R.S. Sutton, and C.W. Anderson. Neuronlike elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, 13:835–846, 1983.

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

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

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

[7] S. Bhatnagar and J. R. Panigrahi. Actor-critic algorithms for hierarchical Markov decision processes. *Automatica*, 42:637–644, 2006.

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

[9] S.J. Bradtke and M. Duff. Reinforcement learning methods for continuous-time Markov decision problems. In *Advances in Neural Information Processing Systems 7*. MIT Press, Cambridge, MA, 1995.

[10] R.I. Brafman and M. Tennenholtz. R-max: A general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3:213–231, 2002.

[11] G.K. Chan and S. Asgarpoor. Optimal maintenance policy with Markov processes. *Electric Power Systems Research*, 76:452–456, 2006.

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

[13] D. Chen and K.S. Trivedi. Optimization for cindition-based maintenance with semi-Markov decision process. *Reliability Engineering and System Safety*, 90:25–29, 2005.

[14] T. K. Das and S. Sarkar. Optimal preventive maintenance in a production inventory system. *IIE Transactions on Quality and Reliability*, 31. (in press).

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

[16] C. Diuk, L. Li, and B.R. Leffler. The adaptive k-meteorologists problem and its application to structure learning and feature selection in reinforcement learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, 2009.

[17] 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)*. John Wiley and Sons, New York, NY, USA, 2005.

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

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

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

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

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

[23] A. Gosavi. Target-sensitive control of Markov and semi-Markov processes. To appear in *International Journal of Control, Automation, and Systems*, 2011.

[24] A. Gosavi, S. Murray, and J. Hu. Model-building semi-Markov adaptive critics. In *Proceedings of the IEEE Symposium on Computational Intelligence: ADPRL, Paris, France*, 2011.

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

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

[27] M. Kearns and S. Singh. Near-optimal reinforcement learning in polynomial time. *Machine Learning*, 49(2):209–232, 2002.

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

[29] R. Koppejan and S. Whiteson. Neuroevolutionary reinforcement learning for generalized helicopter control. In *GECCO: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 145–152, 2009.

[30] K. Kulkarni, A. Gosavi, S. Murray, and K. Grantham. Semi-Markov adpative critic heuristics with application to airline revenue management. *Journal of Control Theory and Applications (Special issue on Reinforcement Learning and Approximate Dynamic Programming)*, 9:421–430, 2011.

[31] K. McKone, R.G. Schhroeder, and K.O. Cua. Total productive maintenance: A contextual review. *Journal of operations management*, 17:123–144, 1999.

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

[33] A.W. Moore and C.G. Atkeson. Prioritized sweeping: Reinforcement learning with less data and less real time. *Machine Learning*, 13:103–130, 1993.

[34] 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.

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

[36] J.R. 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*, pages 330–335. IEEE, 2007.

[37] H. Robbins and S. Monro. A stochasstic approximation method. *Ann. Math. Statist.*, 22:400–407, 1951.

[38] C-R. Robelin and S.M. Madanat. History-dependent bridge deck maintenance and replacement optimization with Markov decision processes. *Journal of Infrastructure Systems*, 13(3):195–201, 2007.

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

[40] A.L. Strehl and M.L. Littman. A theoretical analysis of model-based interval estimation. In *Proceedings of the 22th International Conference on Machine Learning*, page 856863, 2005.

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

[42] R.S. Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the 7th International Workshop on Machine Learning*, pages 216–224. Morgan Kaufmann, San Mateo, CA, 1990.

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

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

[45] H. van Seijen, S. Whiteson, H. van Hasselt, and M. Wiering. Exploiting best-match equations for efficient reinforcement learning. *Journal of Machine Learning Research*, 12:2045–2094, 2011.

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

[47] 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*, 17:7–20, 1987.

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

[49] 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.

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

APPENDIX

$\mathbf{P}_x$=

$$\begin{bmatrix}
.10 & .12 & .10 & .10 & .13 & .05 & .11 & .12 & .10 & .07 \\
.15 & .20 & .05 & .05 & .20 & .10 & .08 & .06 & .06 & .05 \\
.08 & .10 & .20 & .20 & .09 & .05 & .10 & .13 & .02 & .03 \\
.06 & .07 & .10 & .02 & .10 & .20 & .24 & .05 & .04 & .12 \\
.15 & .15 & .04 & .08 & .06 & .08 & .20 & .20 & .02 & .02 \\
.03 & .02 & .10 & .30 & .10 & .10 & .17 & .05 & .10 & .03 \\
.05 & .03 & .10 & .10 & .20 & .15 & .13 & .10 & .04 & .10 \\
.20 & .11 & .05 & .05 & .09 & .10 & .10 & .10 & .10 & .10 \\
.03 & .03 & .10 & .20 & .15 & .15 & .04 & .08 & .16 & .06 \\
.11 & .10 & .10 & .20 & .05 & .12 & .09 & .09 & .10 & .04
\end{bmatrix}$$

$\mathbf{P}_y$=

$$\begin{bmatrix}
.10 & .20 & .07 & .10 & .03 & .20 & .10 & .06 & .10 & .04 \\
.05 & .06 & .10 & .08 & .20 & .11 & .08 & .12 & .02 & .18 \\
.10 & .09 & .10 & .18 & .20 & .10 & .03 & .10 & .04 & .06 \\
.12 & .13 & .02 & .15 & .15 & .12 & .10 & .10 & .05 & .06 \\
.08 & .03 & .18 & .20 & .09 & .10 & .15 & .02 & .07 & .08 \\
.10 & .10 & .10 & .10 & .10 & .10 & .20 & .10 & .09 & .01 \\
.15 & .10 & .08 & .10 & .10 & .15 & .11 & .11 & .06 & .04 \\
.06 & .10 & .10 & .10 & .20 & .06 & .10 & .20 & .04 & .04 \\
.12 & .02 & .07 & .07 & .10 & .20 & .20 & .01 & .02 & .19 \\
.10 & .10 & .23 & .10 & .10 & .07 & .05 & .15 & .01 & .09
\end{bmatrix}$$

$\mathbf{R}_x$=

$$\begin{bmatrix}
-25 & 5 & 50 & -20 & -30 & 10 & -5 & 12 & -40 & 2 \\
10 & 2 & 13 & 26 & 5 & 11 & 12 & 8 & 30 & 9 \\
9 & 5 & 3 & -12 & 5 & 23 & 10 & 18 & 4 & 19 \\
12 & 3 & 22 & 15 & 5 & 32 & -4 & 10 & 15 & 6 \\
28 & 3 & -32 & 20 & 9 & -23 & 12 & 17 & 17 & 8 \\
22 & 4 & 6 & 12 & 12 & 2 & -18 & 18 & 19 & 5 \\
15 & 45 & 17 & -37 & 10 & 9 & 8 & -12 & -6 & 16 \\
6 & -15 & 12 & 21 & -34 & 6 & 3 & 11 & 12 & 4 \\
-23 & 2 & 7 & 17 & 15 & -25 & 19 & -29 & -2 & 9 \\
-20 & 7 & 17 & 5 & 12 & -47 & 5 & 5 & -61 & 69
\end{bmatrix}$$

$\mathbf{R}_y$=

$$\begin{bmatrix}
11 & 20 & 12 & -2 & -12 & 20 & -10 & 16 & -23 & -4 \\
15 & -6 & 20 & -8 & 2 & 11 & 18 & 1 & -22 & 18 \\
-2 & 12 & 12 & 13 & 2 & 12 & 23 & 18 & -14 & 22 \\
12 & 53 & 22 & -15 & -45 & 22 & 10 & 90 & 12 & 6 \\
-28 & 3 & 38 & -20 & 39 & -10 & -7 & 2 & 47 & -38 \\
-12 & 14 & 66 & 42 & 45 & -34 & -18 & 58 & 39 & 61 \\
35 & 9 & 47 & -10 & -20 & 49 & -8 & 23 & 36 & 46 \\
-6 & -40 & 10 & 60 & -20 & -6 & 50 & 20 & 54 & 4 \\
33 & 2 & -37 & 47 & -45 & -25 & 39 & 1 & 42 & 19 \\
10 & 7 & -26 & 10 & 20 & 27 & -35 & -15 & 1 & 9
\end{bmatrix}$$

$\mathbf{T}_x =$

$$= \begin{bmatrix} 3 & 5 & 30 & 20 & 10 & 13 & 54 & 12 & 40 & 12 \\ 100 & 2 & 131 & 46 & 5 & 11 & 17 & 48 & 10 & 29 \\ 49 & 54 & 3 & 12 & 25 & 45 & 40 & 28 & 34 & 8 \\ 32 & 23 & 2 & 35 & 25 & 62 & 4 & 1 & 150 & 6 \\ 28 & 23 & 9 & 10 & 29 & 23 & 2 & 14 & 19 & 18 \\ 22 & 4 & 26 & 120 & 128 & 29 & 10 & 108 & 119 & 22 \\ 35 & 6 & 37 & 89 & 30 & 29 & 48 & 52 & 6 & 36 \\ 6 & 25 & 52 & 32 & 34 & 60 & 30 & 110 & 132 & 64 \\ 23 & 2 & 47 & 17 & 15 & 75 & 10 & 90 & 72 & 90 \\ 44 & 47 & 67 & 95 & 12 & 4 & 55 & 15 & 17 & 9 \end{bmatrix}$$

$\mathbf{T}_y =$

$$\begin{bmatrix} 110 & 30 & 4 & 25 & 120 & 70 & 10 & 10 & 25 & 34 \\ 55 & 26 & 200 & 98 & 2 & 11 & 18 & 1 & 82 & 8 \\ 2 & 12 & 24 & 1 & 20 & 82 & 3 & 68 & 14 & 67 \\ 32 & 13 & 42 & 45 & 15 & 22 & 100 & 90 & 10 & 56 \\ 8 & 33 & 30 & 25 & 39 & 14 & 47 & 21 & 7 & 18 \\ 70 & 4 & 36 & 52 & 45 & 12 & 48 & 8 & 19 & 91 \\ 55 & 29 & 7 & 5 & 10 & 29 & 28 & 90 & 96 & 86 \\ 46 & 20 & 10 & 60 & 24 & 16 & 10 & 10 & 54 & 34 \\ 3 & 42 & 23 & 97 & 5 & 65 & 10 & 16 & 32 & 79 \\ 10 & 37 & 6 & 60 & 60 & 97 & 95 & 5 & 16 & 89 \end{bmatrix}$$

Table 7. Parameters for Cases 2 through 10. Parameters not defined are identical to those for Case 1.

Case 2:
$p(4,1,7) = .20, p(4,1,8) = .09, p(7,1,6) = .17, p(7,1,7) = .11, p(9,1,5) = .10$ $p(9,1,6) = .20, p(5,2,2) = .09, p(5,2,3) = .12, p(7,2,6) = .12, p(7,2,7) = .14$ $p(9,2,1) = .10, p(9,2,3) = .09$ $r(2,1,8) = 28, r(3,1,7) = 5, r(4,1,9) = 50, r(5,1,9) = -17, r(10,1,7) = 25$ $r(2,2,7) = 21, r(3,2,6) = -20, r(4,2,9) = -20$ $t(4,1,8) = 70, t(7,1,1) = 5, t(1,2,4) = 75, t(9,2,2) = 12$
Case 3:
$p(4,1,7) = .20, p(4,1,8) = .09, p(7,1,6) = .17, p(7,1,7) = .11, p(9,1,5) = .10$ $p(9,1,6) = .20, p(5,2,2) = .09, p(5,2,3) = .12, p(7,2,6) = .12, p(7,2,7) = .14$ $p(9,2,1) = .10, p(9,2,3) = .09$ $r(1,1,2) = -5, r(2,1,8) = 28, r(3,1,7) = 5, r(4,1,9) = 50, r(5,1,9) = -17$ $r(8,1,8) = 41, r(10,1,7) = 25, r(1,2,1) = -11, r(1,2,2) = -20, r(2,2,7) = 21$ $r(3,2,6) = -20, r(4,2,9) = -20, r(7,2,9) = 16$ $t(1,1,2) = 50, t(4,1,7) = 24, t(4,1,8) = 70, t(5,1,6) = 63, t(7,1,1) = 5$ $t(10,1,8) = 45, t(1,2,3) = 40, t(1,2,4) = 75, t(9,2,2) = 12, t(10,2,8) = 55$
Case 4:
$p(4,1,7) = .20, p(4,1,8) = .09, p(7,1,6) = .17, p(7,1,7) = .11, p(9,1,5) = .10$ $p(9,1,6) = .20, p(5,2,2) = .09, p(5,2,3) = .12, p(7,2,6) = .12, p(7,2,7) = .14$ $p(9,2,1) = .10, p(9,2,3) = .09$ $r(1,1,2) = -5, r(2,1,8) = 28, r(3,1,7) = 5, r(4,1,9) = 50, r(5,1,9) = -17$ $r(7,1,4) = 37, r(7,1,7) = 28, r(8,1,1) = 5, r(8,1,2) = 15, r(8,1,3) = 20$ $r(8,1,8) = 41, r(9,1,2) = 22, r(9,1,3) = 47, r(10,1,1) = 20, r(10,1,2) = 37$ $r(10,1,7) = 25, r(1,2,1) = -11, r(1,2,2) = -20, r(2,2,7) = 21, r(3,2,6) = -20$ $r(4,2,9) = -20, r(7,2,2) = -9, r(7,2,6) = -49, r(7,2,7) = 8, r(7,2,9) = 16$ $r(8,2,1) = 6, r(8,2,5) = 20, r(10,2,2) = -7$ $t(1,1,2) = 50, t(4,1,7) = 24, t(4,1,8) = 70, t(5,1,6) = 63, t(7,1,1) = 5$ $t(10,1,8) = 45, t(1,2,3) = 40, t(1,2,4) = 75, t(9,2,2) = 12, t(10,2,8) = 55$
Case 5:
$r(7,1,3) = 90, r(7,1,7) = 80, r(10,1,1) = 60, r(10,1,4) = 35, r(10,1,6) = -17$ $r(10,1,8) = 65, r(10,2,2) = -9, r(10,2,1) = 50, r(10,2,3) = -6, r(10,2,7) = -15$ $r(10,2,8) = -5$
Case 6:
$p(7,1,1) = .09, p(7,1,6) = .11, p(9,2,9) = .09, p(9,2,10) = .12, p(10,2,6) = .09$ $p(10,2,8) = .13$ $r(2,1,1) = -10, r(2,1,4) = -46, r(2,1,8) = -18$ $t(6,1,3) = 70, t(9,1,5) = 55, t(3,2,8) = 28, t(6,2,5) = 15, t(7,2,9) = 46$
Case 7:
$p(4,1,7) = .20, p(4,1,8) = .09, p(4,2,2) = .16, p(4,2,4) = .12$ $r(3,1,2) = 45, r(3,1,3) = 23, r(4,1,5) = 35, r(7,1,3) = 67, r(6,2,3) = 16$ $r(7,2,2) = 29$
Case 8:
$r(5,1,3) = -92, r(5,2,1) = 58, r(5,2,3) = 88$ $t(1,1,2) = 55, t(6,2,6) = 72$
Case 9:
$r(5,1,5) = 69, t(5,2,4) = 50, t(6,2,3) = 16$ $t(3,1,2) = 4, t(1,2,3) = 74$
Case 10:
$r(10,1,1) = 80, r(10,1,5) = 82, r(10,1,8) = 35, r(10,1,9) = 81, r(10,2,1) = 40$ $r(10,2,2) = 37, r(10,2,3) = 56, r(10,2,9) = 63, r(10,2,10) = 21$ $t(9,1,2) = 57, t(10,1,1) = 14, t(10,1,4) = 15, t(3,2,4) = 60, t(9,2,7) = 80$