Machine Learning, 22, 123–158 (1996)
© 1996 Kluwer Academic Publishers, Boston. Manufactured in The Netherlands.

Reinforcement Learning with Replacing Eligibility Traces

SATINDER P. SINGH singh@psyche.mit.edu
Dept. of Brain and Cognitive Sciences
Massachusetts Institute of Technology, Cambridge, Mass. 02139

RICHARD S. SUTTON rich@cs.umass.edu
Dept. of Computer Science
University of Massachusetts, Amherst, Mass. 01003

Editor: Leslie Pack Kaelbling

Abstract. The eligibility trace is one of the basic mechanisms used in reinforcement learning to handle delayed reward. In this paper we introduce a new kind of eligibility trace, the *replacing* trace, analyze it theoretically, and show that it results in faster, more reliable learning than the conventional trace. Both kinds of trace assign credit to prior events according to how recently they occurred, but only the conventional trace gives greater credit to repeated events. Our analysis is for conventional and replace-trace versions of the offline TD(1) algorithm applied to undiscounted absorbing Markov chains. First, we show that these methods converge under repeated presentations of the training set to the same predictions as two well known Monte Carlo methods. We then analyze the relative efficiency of the two Monte Carlo methods. We show that the method corresponding to conventional TD is biased, whereas the method corresponding to replace-trace TD is unbiased. In addition, we show that the method corresponding to replacing traces is closely related to the maximum likelihood solution for these tasks, and that its mean squared error is always lower in the long run. Computational results confirm these analyses and show that they are applicable more generally. In particular, we show that replacing traces significantly improve performance and reduce parameter sensitivity on the "Mountain-Car" task, a full reinforcement-learning problem with a continuous state space, when using a feature-based function approximator.

Keywords: reinforcement learning, temporal difference learning, eligibility trace, Monte Carlo method, Markov chain, CMAC

1. Eligibility Traces

Two fundamental mechanisms have been used in reinforcement learning to handle delayed reward. One is temporal-difference (TD) learning, as in the TD(λ) algorithm (Sutton, 1988) and in Q-learning (Watkins, 1989). TD learning in effect constructs an internal reward signal that is less delayed than the original, external one. However, TD methods can eliminate the delay completely only on fully Markov problems, which are rare in practice. In most problems some delay always remains between an action and its effective reward, and on all problems some delay is always present during the time before TD learning is complete. Thus, there is a general need for a second mechanism to handle whatever delay is not eliminated by TD learning.

The second mechanism that has been widely used for handling delay is the *eligibility trace*.[1] Introduced by Klopf (1972), eligibility traces have been used in a variety of reinforcement learning systems (e.g., Barto, Sutton & Anderson, 1983; Lin, 1992; Tesauro, 1992; Peng & Williams, 1994). Systematic empirical studies of eligibility traces in conjunction with TD methods were made by Sutton (1984), and theoretical results have been obtained by several authors (e.g., Dayan, 1992; Jaakkola, Jordan & Singh, 1994; Tsitsiklis, 1994; Dayan & Sejnowski, 1994; Sutton & Singh, 1994).

The idea behind all eligibility traces is very simple. Each time a state is visited it initiates a short-term memory process, a trace, which then decays gradually over time. This trace marks the state as *eligible* for learning. If an unexpectedly good or bad event occurs while the trace is non-zero, then the state is assigned credit accordingly. In a conventional *accumulating trace*, the trace builds up each time the state is entered. In a *replacing trace*, on the other hand, each time the state is visited the trace is reset to 1 regardless of the presence of a prior trace. The new trace replaces the old. See Figure 1.

Sutton (1984) describes the conventional trace as implementing the credit assignment heuristics of *recency*—more credit to more recent events—and *frequency*—more credit to events that have occurred more times. The new replacing trace can be seen simply as discarding the frequency heuristic while retaining the recency heuristic. As we show later, this simple change can have a significant effect on performance.

Typically, eligibility traces decay exponentially according to the product of a decay parameter, λ, $0 \leq \lambda \leq 1$, and a discount-rate parameter, γ, $0 \leq \gamma \leq 1$. The conventional accumulating trace is defined by:[2]

$$e_{t+1}(s) = \begin{cases} \gamma\lambda e_t(s) & \text{if } s \neq s_t; \\ \gamma\lambda e_t(s) + 1 & \text{if } s = s_t, \end{cases}$$

where $e_t(s)$ represents the trace for state s at time t, and s_t is the actual state at time t. The corresponding replacing trace is defined by:

$$e_{t+1}(s) = \begin{cases} \gamma\lambda e_t(s) & \text{if } s \neq s_t; \\ 1 & \text{if } s = s_t. \end{cases}$$

In a control problem, each state-action *pair* has a separate trace. When a state is visited and an action taken, the state's trace for that action is reset to 1 while the traces for the other actions are reset to zero (see Section 5).

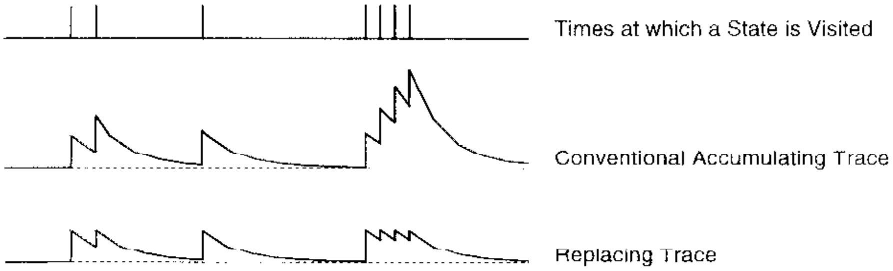
Times at which a State is Visited
Conventional Accumulating Trace
Replacing Trace

Figure 1. Accumulating and replacing eligibility traces.

For problems with a large state space it may be extremely unlikely for the exact same state ever to recur, and thus one might think replacing traces would be irrelevant. However, large problems require some sort of generalization between states, and thus some form of function approximator. Even if the same *states* never recur, states with the same *features* will. In Section 5 we show that replacing traces do indeed make a significant difference on problems with a large state space when the traces are done on a feature-by-feature basis rather than on a state-by-state basis.

The rest of this paper is structured as follows. In the next section we review the TD(λ) prediction algorithm and prove that its variations using accumulating and replacing traces are closely related to two Monte Carlo algorithms. In Section 3 we present our main results on the relative efficiency of the two Monte Carlo algorithms. Sections 4 and 5 are empirical and return to the general case.

2. TD(λ) and Monte Carlo Prediction Methods

The prediction problem we consider is a classical one in reinforcement learning and optimal control. A Markov chain emits on each of its transitions a *reward*, $r_{t+1} \in \Re$, according to a probability distribution dependent only on the pre-transition state, s_t, and the post-transition state, s_{t+1}. For each state, we seek to predict the expected total (cumulative) reward emitted starting from that state until the chain reaches a terminal state. This is called the *value* of the state, and the function mapping states s to their values $V(s)$ is called the *value function*. In this paper, we assume no discounting ($\gamma = 1$) and that the Markov chain always reaches a terminal state. Without loss of generality we assume that there is a single terminal state, T, with value $V(T) = 0$. A single trip from starting state to terminal state is called a *trial*.

2.1. TD(λ) Algorithms

The TD(λ) family of algorithms combine TD learning with eligibility traces to estimate the value function. The discrete-state form of the TD(λ) algorithm is defined by

$$\Delta V_t(s) = \alpha_t(s)\Big[r_{t+1} + V_t(s_{t+1}) - V_t(s_t)\Big]e_{t+1}(s) \qquad \forall s, \forall t \text{ s.t. } s_t \neq T, \tag{1}$$

where $V_t(s)$ is the estimate at time t of $V(s)$, $\alpha_t(s)$ is a positive step-size parameter, $e_{t+1}(s)$ is the eligibility trace for state s, and $\Delta V_t(s)$ is the increment in the estimate of $V(s)$ determined at time t.[3] The value at the terminal state is of course defined as $V_t(T) = 0$, $\forall t$. In *online* TD(λ), the estimates are incremented on every time step: $V_{t+1}(s) = V_t(s) + \Delta V_t(s)$. In *offline* TD($\lambda$), on the other hand, the increments $\Delta V_t(s)$ are set aside until the terminal state is reached. In this case the estimates $V_t(s)$ are constant while the chain is undergoing state transitions, all changes being deferred until the end of the trial.

There is also a third case in which updates are deferred until after an entire set of trials have been presented. Usually this is done with a small fixed step size, $\alpha_t(s) = \alpha$, and with the training set (the set of trials) presented over and over again until convergence of the value estimates. Although this "repeated presentations" training paradigm is rarely used in practice, it can reveal telling theoretical properties of the algorithms. For example, Sutton (1988) showed that TD(0) (TD(λ) with $\lambda = 0$) converges under these conditions to a maximum likelihood estimate, arguably the best possible solution to this prediction problem (see Section 2.3). In this paper, for convenience, we refer to the repeated presentations training paradigm simply as *batch* updating. Later in this section we show that the batch versions of conventional and replace-trace TD(1) methods are equivalent to two Monte Carlo prediction methods.

2.2. *Monte Carlo Algorithms*

The total reward following a particular visit to a state is called the *return* for that visit. The value of a state is thus the expected return. This suggests that one might estimate a state's value simply by averaging all the returns that follow it. This is what is classically done in *Monte Carlo* (MC) prediction methods (Rubinstein, 1981; Curtiss, 1954; Wasow, 1952; Barto & Duff, 1994). We distinguish two specific algorithms:

Every-visit MC: Estimate the value of a state as the average of the returns that have followed all visits to the state.

First-visit MC: Estimate the value of a state as the average of the returns that have followed the *first visits* to the state, where a first visit is the first time during a trial that the state is visited.

Note that both algorithms form their estimates based entirely on actual, complete returns. This is in contrast to TD(λ), whose updates (1) are based in part on existing estimates. However, this is only in part, and, as $\lambda \to 1$, TD(λ) methods come to more and more closely approximate MC methods (Sections 2.4 and 2.5). In particular, the conventional, accumulate-trace version of TD(λ) comes to approximate every-visit MC, whereas replace-trace TD(λ) comes to approximate first-visit MC. One of the main points of this paper is that we can better understand the difference between replace and accumulate versions of TD(λ) by understanding the difference between these two MC methods. This naturally brings up the question that we focus on in Section 3: what are the relative merits of first-visit and every-visit MC methods?

2.3. *A Simple Example*

To help develop intuitions, first consider the very simple Markov chain shown in Figure 2a. On each step, the chain either stays in S with probability p, or goes on to terminate in T with probability $1 - p$. Suppose we wish to estimate the expected number of steps before termination when starting in S. To put this in the form of estimating a value function, we say that a reward of $+1$ is emitted on every step, in which case $V(S)$

is equal to the expected number of steps before termination. Suppose that the only data that has been observed is a single trial generated by the Markov chain, and that that trial lasted 4 steps, 3 passing from S to S, and one passing from S to T, as shown in Figure 2b. What do the two MC methods conclude from this one trial?

We assume that the methods do not know the structure of the chain. All they know is the one experience shown in Figure 2b. The first visit MC method in effect sees a single traversal from the first time S was visited to T. That traversal lasted 4 steps, so its estimate of $V(S)$ is 4. Every-visit MC, on the other hand, in effect sees 4 separate traversals from S to T, one with 4 steps, one with 3 steps, one with 2 steps, and one with 1 step. Averaging over these four effective trials, every-visit MC estimates $V(S)$ as $\frac{1+2+3+4}{4} = 2.5$. The replace and accumulate versions of TD(1) may or may not form exactly these estimates, depending on their α sequence, but they will move their estimates in these directions. In particular, if the corresponding offline TD(1) method starts the trial with these estimates, then it will leave them unchanged after experiencing the trial. The batch version of the two TD(1) algorithms will compute exactly these estimates.

Which estimate is better, 4 or 2.5? Intuitively, the first answer appears better. The only trial observed took 4 steps, so 4 seems like the best estimate of its expected value. In any event, the answer 2.5 seems too low. In a sense, the whole point of this paper is to present theoretical and empirical analyses in support of this intuition. We show below that in fact the answer 4 is the only unbiased answer, and that 2.5, the answer of every-visit MC and of conventional TD(1), is biased in a statistical sense.

It is instructive to compare these two estimates of the value function with the estimate that is optimal in the maximum likelihood sense. Given some data, in this case a set of observed trials, we can construct the maximum-likelihood model of the underlying Markov process. In general, this is the model whose probability of generating the observed data is the highest. Consider our simple example. After the one trial has been observed, the maximum-likelihood estimate of the S-to-S transition probability is $\frac{3}{4}$, the fraction of the actual transitions that went that way, and the maximum-likelihood estimate of the S to T transition probability is $\frac{1}{4}$. No other transitions have been observed, so they are estimated as having probability 0. Thus, the maximum-likelihood model of the Markov chain is as shown in Figure 2c.

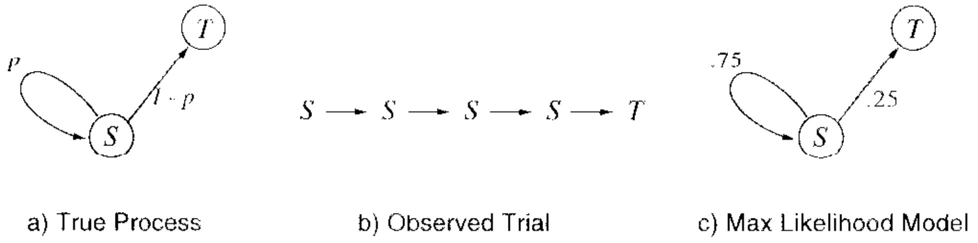

Figure 2. A simple example of a Markov prediction problem. The objective is to predict the number of steps until termination.

We define the *ML estimate* of the value function as the value function that would be exactly correct if the maximum-likelihood model of the Markov process were exactly correct. That is, it is the estimate equal to the correct answer if the estimate of the Markov chain was not really an estimate, but was known with certainty.[4] Note that the ML estimate makes full use of all the observed data.

Let us compute the ML estimate for our simple example. If the maximum-likelihood model of the chain, as shown in Figure 2c, were exactly correct, what then would be the expected number of time steps before termination? For each possible number of steps, k, we can compute the probability of its occurring, and then the expected number, as

$$\begin{aligned} V^{ML}(S) &= \sum_{k=1}^{\infty} \Pr(k)k \\ &= \sum_{k=1}^{\infty} (0.75)^{k-1} \cdot 0.25 \cdot k \\ &= 4. \end{aligned}$$

Thus, in this simple example the ML estimate is the same as the first-visit MC estimate. In general, these two are not exactly the same, but they are closely related. We establish the relationship in the general case in Section 3.2.

Computing the ML estimate is in general very computationally complex. If the number of states is n, then the maximum-likelihood model of the Markov chain requires $O(n^2)$ memory, and computing the ML estimate from it requires roughly $O(n^3)$ computational operations.[5] The TD methods by contrast all use memory and computation per step that is only $O(n)$. It is in part because of these computational considerations that learning solutions are of interest while the ML estimate remains an ideal generally unreachable in practice. However, we can still ask how closely the various learning methods approximate this ideal.

2.4. *Equivalence of Batch TD(1) and MC Methods*

In this subsection we establish that the replace and accumulate forms of batch TD(1) are equivalent, respectively, to first-visit and every-visit MC. The next subsection proves a similar equivalence for the offline TD(1) algorithms.

The equivalence of the accumulate-trace version of batch TD(1) to every-visit MC follows immediately from prior results. Batch TD(1) is a gradient-descent procedure known to converge to the estimate with minimum mean squared error on the training set (Sutton, 1988; Dayan, 1992; Barnard, 1993). In the case of discrete states, the minimum MSE estimate for a state is the sample average of the returns from every visit to that state in the training set, and thus it is the same as the estimate computed by every-visit MC.

Showing the equivalence of replace-trace batch TD(1) and first-visit MC requires a little more work.

THEOREM 1: *For any training set of N trials and any fixed $\alpha_t(s) = \alpha < \frac{1}{N}$, batch replace TD(1) produces the same estimates as first-visit MC.*

Proof: In considering updates to the estimates for any state s we need only consider trials in which s occurs. On trials in which s does not occur, the estimates of both methods are obviously unchanged. We index the trials in which state s occurs from 1 to $N(s)$. Let $t_s(n)$ be the time at which state s is first visited in trial n, and let $t_T(n)$ be the time at which the terminal state is reached. Let $V_i^R(s)$ represent the replace TD(1) estimate of the value of state s after i passes through the training set, for $i \geq 1$:

$$\begin{aligned}
V_{i+1}^R(s) &= V_i^R(s) + \sum_{n=1}^{N(s)} \sum_{t=t_s(n)}^{t_T(n)-1} \Delta V_t(s) \\
&= V_i^R(s) + \alpha \sum_{n=1}^{N(s)} \sum_{t=t_s(n)}^{t_T(n)-1} \left[r_{t+1} + V_i^R(s_{t+1}) - V_i^R(s_t)\right] \\
&= V_i^R(s) + \alpha \sum_{n=1}^{N(s)} \left[-V_i^R(s_{t_s(n)}) + \sum_{t=t_s(n)}^{t_T(n)-1} r_{t+1} \right] \\
&= V_i^R(s) + \alpha \sum_{n=1}^{N(s)} \left[R(t_s(n)) - V_i^R(s)\right] \\
&= (1 - N(s)\alpha)V_i^R(s) + \alpha \sum_{n=1}^{N(s)} R(t_s(n)),
\end{aligned}$$

where $R(t)$ is the return following time t to the end of the trial. This in turn implies that

$$V_i^R(s) = (1 - N(s)\alpha)^i V_0^R(s) + \alpha \sum_{n=1}^{N(s)} R(t_s(n)) \left[1 + (1 - N(s)\alpha) + \ldots (1 - N(s)\alpha)^{i-1}\right].$$

Therefore,

$$\begin{aligned}
V_\infty^R(s) &= (1 - N(s)\alpha)^\infty V_0^R(s) + \alpha \sum_{n=1}^{N(s)} R(t_s(n)) \sum_{j=0}^{\infty} (1 - N(s)\alpha)^j \\
&= \alpha \sum_{n=1}^{N(s)} R(t_s(n)) \frac{1}{1 - (1 - N(s)\alpha)} \qquad \text{(because } N(s)\alpha < 1\text{)} \\
&= \frac{\sum_{n=1}^{N(s)} R(t_s(n))}{N(s)},
\end{aligned}$$

which is the first-visit MC estimate. ■

2.5. *Equivalence of Offline TD(1) and MC Methods by Choice of α*

In this subsection we establish that the replace and accumulate forms of *offline* TD(1) can also be made equivalent to the corresponding MC methods by suitable choice of the step-size sequence $\alpha_t(s)$.

THEOREM 2: *Offline replace TD(1) is equivalent to first-visit MC under the step-size schedule*

$$\alpha_t(s) = \frac{1}{\textit{number of first visits to } s \textit{ up through time } t}.$$

Proof: As before, in considering updates to the estimates of $V(s)$, we need only consider trials in which s occurs. The cumulative increment in the estimate of $V(s)$ as a result of the i^{th} trial in which s occurs is

$$\begin{aligned}\sum_{t=t_s(i)}^{t_T(i)} \Delta V_t(s) &= \sum_{t=t_s(i)}^{t_T(i)} \alpha_t(s)\left[r_{t+1} + V_{i-1}^R(s_{t+1}) - V_{i-1}^R(s_t)\right] \\ &= \frac{1}{i}(R(t_s(i)) - V_{i-1}^R(s)).\end{aligned}$$

Therefore, the update for offline replace TD(1), after a complete trial, is

$$V_i^R(s) = V_{i-1}^R(s) + \frac{1}{i}(R(t_s(i)) - V_{i-1}^R(s)),$$

which is just the iterative recursive equation for incrementally computing the average of the first-visit returns, $\{R(t_s(1)), R(t_s(2)), R(t_s(3)), \ldots\}$. ■

THEOREM 3: *Offline accumulate TD(1) is equivalent to every-visit MC under the step-size schedule*

$$\alpha_t(s) = \frac{1}{\textit{number of visits to } s \textit{ up through the entire trial containing time } t}$$

Proof: Once again we consider only trials in which state s occurs. For this proof we need to use the time index of every visit to state s, complicating notation somewhat. Let $t_s(i;k)$ be the time index of the k^{th} visit to state s in trial i. Also, let $K_s(i)$ be the total number of visits to state s in trial i. The essential idea behind the proof is to again show that the offline TD(1) equation is an iterative recursive averaging equation, only this time of the returns from every visit to state s.

Let $\alpha_i(s)$ be the step-size parameter used in processing trial i. The cumulative increment in the estimate of $V(s)$ as a result of trial i is

$$\sum_{t=t_s(i;1)}^{t_T(i)} \Delta V_t(s) = \alpha_i(s)\left[\sum_{t=t_s(i;1)}^{t_s(i;2)-1} \Delta_{i-1}(s_t) + 2\sum_{t=t_s(i;2)}^{t_s(i;3)-1} \Delta_{i-1}(s_t)\right.$$

$$+ \cdots + K_s(i) \sum_{t=t_s(i;K_s(i))}^{t_T(i)} \Delta_{i-1}(s_t)\Bigg]$$

$$= \alpha_i(s) \left[\sum_{j=1}^{K_s(i)} R(t_s(i;j)) - K_s(i) V_{i-1}^A(s) \right],$$

where $\Delta_i(s_t) = r_{t+1} + V_i^A(s_{t+1}) - V_i^A(s_t)$, and $V_i^A(s)$ is the accumulate-trace estimate at trial i. Therefore,

$$V_i^A(s) = V_{i-1}^A(s) + \alpha_i(s) \left[\sum_{j=1}^{K_s(i)} R(t_s(i;j)) - K_s(i) V_{i-1}^A(s) \right].$$

Because $\alpha_i(s) = \frac{1}{\sum_{j=1}^{i} K_s(j)}$, this will compute the sample average of all the actual returns from every visit to state s up to and including trial i. ■

3. Analytic Comparison of Monte Carlo Methods

In the previous section we established close relationships of replace and accumulate TD(1) to first-visit and every-visit MC methods respectively. By better understanding the difference between the MC methods, then, we might hope to better understand the difference between the TD methods. Accordingly, in this section we evaluate analytically the quality of the solutions found by the two MC methods. In brief, we explore the asymptotic correctness of all methods, the bias of the MC methods, the variance and mean-squared error of the MC methods, and the relationship of the MC methods to the maximum-likelihood estimate. The results of this section are summarized in Table 1.

3.1. Asymptotic Convergence

In this subsection we briefly establish the asymptotic correctness of the TD methods. The asymptotic convergence of accumulate TD(λ) for general λ is well known (Dayan, 1992; Jaakkola, Jordan & Singh, 1994; Tsitsiklis, 1994; Peng, 1993). The main results appear to carry over to the replace-trace case with minimal modifications. In particular:

THEOREM 4: *Offline (online) replace TD(λ) converges to the desired value function w.p.1 under the conditions for w.p.1 convergence of offline (online) conventional TD(λ) stated by Jaakkola, Jordan and Singh (1994).*

Proof: Jaakkola, Jordan and Singh (1994) proved that online and offline TD(λ) converges $w.p.1$ to the correct predictions, under natural conditions, as the number of trials goes to infinity (or as the number of time steps goes to infinity in the online, non-absorbing case, with $\gamma < 1$). Their proof is based on showing that the offline TD(λ) estimator is a contraction mapping in expected value. They show that it is a weighted sum of n-step corrected truncated returns,

$$V_t^{(n)}(s_t) = r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^{n-1} r_{t+n} + \gamma^n V_t(s_{t+n}),$$

that, for all $n \geq 1$, are better estimates (in expected value) of $V(s_t)$ than is $V_t(s_t)$. The eligibility trace collects successive n-step estimators, and its magnitude determines their weighting. The TD(λ) estimator is

$$\sum_{k=0}^{\tau-1}\Big[r_{t+k+1} + \gamma V_t(s_{t+k+1}) - V_t(s_{t+k})\Big] e_{t+k+1}(s_t) + V_t(s_t) =$$

$$(1-\lambda)\left[\sum_{n=1}^{\tau} \lambda^{n-1} V_t^{(n)}(s_t) + V_t^{(\tau)}(s_t) \sum_{n=\tau+1}^{\infty} \lambda^{n-1}\right],$$

where, for the accumulating trace, τ is the number of time steps until termination, whereas, for the replacing trace, τ is the number of time steps until the next revisit to state s_t. Although the weighted sum is slightly different in the replace case, it is still a contraction mapping in expected value and meets all the conditions of Jaakkola *et al.*'s proofs of convergence for online and offline updating. ■

3.2. Relationship to the ML Estimate

In the simple example in Figure 2, the first-visit MC estimate is the same as the ML estimate. However, this is true in general only for the *starting state*, assuming all trials start in the same state. One way of thinking about this is to consider for any state s just the subset of the training trials that include s. For each of these trials, discard the early part of the trial before s was visited for the first time. Consider the remaining "tails" of the trials as a new training set. This reduced training set is really all the MC methods ever see in forming their estimates for s. We refer to the ML estimate of $V(s)$ based on this reduced training set as the *reduced-ML* estimate. In this subsection we show that the reduced ML estimate is equivalent in general to the first-visit MC estimate.

THEOREM 5: *For any undiscounted absorbing Markov chain, the estimates computed by first-visit MC are the reduced-ML estimates, for all states and after all trials.*

Proof: The first-visit MC estimate is the average of the returns from first visits to state s. Because the maximum-likelihood model is built from the partial experience rooted in state s, the sum over all t of the probability of making a particular transition at time step t according to the maximum-likelihood model is equal to the ratio of the number of times that transition was actually made to the number of trials. Therefore, the reduced-ML estimate for state s is equal to the first-visit MC estimate. See Appendix A.1 for a complete proof. ■

Theorem 5 shows the equivalence of the first-visit MC and reduced-ML estimates. *Every*-visit MC in general produces an estimate different from the reduced-ML estimate.

3.3. Reduction to a Two-State Abstracted Markov Chain

In this subsection we introduce a conceptual reduction of arbitrary undiscounted absorbing Markov chains to a two-state *abstracted Markov chain* that we then use in the rest of this paper's analyses of MC methods. The reduction is based on focusing on each state individually. Assume for the moment that we are interested in the value only of one state, s. We assume that all training trials start in state s. We can do this without loss of generality because the change in the value of a state after a trial is unaffected by anything that happens before the first visit to the state on that trial.

For any Markov chain, a trial produces two sequences, a random sequence of states, $\{s\}$, beginning with s and ending in T, and an associated random sequence of rewards, $\{r\}$. Partition sequence $\{s\}$ into contiguous subsequences that begin in s and end just before the next revisit to s. The subsequence starting with the i^{th} revisit to s is denoted $\{s\}_i$. The last such subsequence is special in that it ends in the terminal state and is denoted $\{s\}_T$. The corresponding reward sequences are similarly denoted $\{r\}_i$ and $\{r\}_T$. Because of the Markov property, $\{s\}_i$ is independent of $\{s\}_j$, for all $i \neq j$, and similarly $\{r\}_i$ is independent of $\{r\}_j$. This is useful because it means that the precise sequence of states that actually occurs between visits to s does not play a role in the first-visit MC or the every-visit MC estimates for $V(s)$. Similarly, the precise sequence of rewards, $\{r\}_i$, does not matter, as only the sum of the rewards in between visits to s are used in the MC methods.

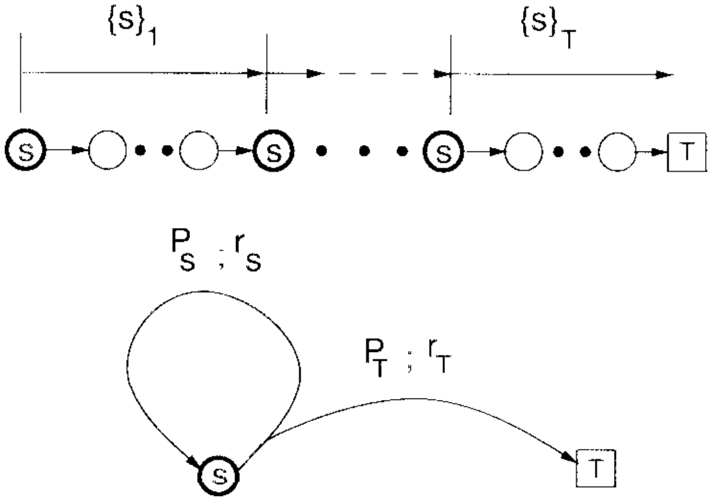

Figure 3. Abstracted Markov chain. At the top is a typical sequence of states comprising a training trial. The sequence can be divided into contiguous subsequences at the visits to start state s. For our analyses, the precise sequence of states and rewards in between revisits to s does not matter. Therefore, in considering the value of s, arbitrary undiscounted Markov chains can be abstracted to the two-state chain shown in the lower part of the figure.

Therefore, for the purpose of analysis, arbitrary undiscounted Markov chains can be reduced to the two-state abstract chain shown in the lower part of Figure 3. The associated probabilities and rewards require careful elaboration. Let P_T and P_s denote the probabilities of terminating and looping respectively in the abstracted chain. Let r_s and r_T represent the random rewards associated with a $s \rightsquigarrow s$ transition and a $s \rightsquigarrow T$ transition in Figure 3. We use the quantities, $R_s = E\{r_s\}$, $Var(r_s) = E\{(r_s - R_s)^2\}$, $R_T = E\{r_T\}$, and $Var(r_T) = E\{(r_T - R_T)^2\}$ in the following analysis. Precise definitions of these quantities are given in Appendix A.2.

first-visit MC:
Let $\{x\}$ stand for the paired random sequence $(\{s\}, \{r\})$. The first-visit MC estimate for $V(s)$ after one trial, $\{x\}$, is

$$V_1^F(s) = f(\{x\}) = r_{s_1} + r_{s_2} + r_{s_3} + \cdots + r_{s_k} + r_T,$$

where k is the *random* number of *revisits* to state s, r_{s_i} is the sum of the individual rewards in the sequence $\{r\}_i$, and r_T is the random total reward received after the last visit to state s. For all i, $E\{r_{s_i}\} = R_s$. The first-visit MC estimate of $V(s)$ after n trials, $\{x\}^1, \{x\}^2, \ldots, \{x\}^n$, is

$$V_n^F(s) = f(\{x\}^1, \{x\}^2, \ldots, \{x\}^n) = \frac{\sum_{i=1}^n f(\{x\}^i)}{n}. \tag{2}$$

In words, $V_n^F(s)$ is simply the average of the estimates from the n sample trajectories, $\{x\}^1, \{x\}^2, \ldots, \{x\}^n$, all of which are independent of each other because of the Markov property.

every-visit MC:
The every-visit MC estimate for one trial, $\{x\}$, is

$$V_1^E(s) = t(\{x\}) = \frac{t_{num}(\{x\})}{k+1} = \frac{r_{s_1} + 2r_{s_2} + \ldots + kr_{s_k} + (k+1)r_T}{k+1},$$

where k is the random number of revisits to state s in the sequence $\{x\}$. Every visit to state s effectively starts another trial. Therefore, the rewards that occur in between the i^{th} and $(i+1)^{st}$ visits to state s are included i times in the estimate.

The every-visit MC estimate after n trials, $\{x\}^1, \{x\}^2, \ldots, \{x\}^n$, is

$$V_n^E(s) = t(\{x\}^1, \{x\}^2, \ldots, \{x\}^n) = \frac{\sum_{i=1}^n t_{num}(\{x\}^i)}{\sum_{i=1}^n (k_i + 1)}, \qquad (3)$$

where k_i is the number of revisits to s in the i^{th} trial $\{x\}^i$. Unlike the first-visit MC estimator, the every-visit MC estimator for n trials is not simply the average of the estimates for individual trials, making its analysis more complex.

We derive the bias ($Bias$) and variance (Var) of first-visit MC and every-visit MC as a function of the number of trials, n. The mean squared error (MSE) is $Bias^2 + Var$.

3.4. *Bias Results*

First consider the true value of state s in Figure 3. From Bellman's equation (Bellman, 1957):

$$V(s) = P_s(R_s + V(s)) + P_T(R_T + V_T)$$

or

$$(1 - P_s)V(s) = P_s R_s + P_T R_T,$$

and therefore

$$V(s) = \frac{P_s}{P_T} R_s + R_T.$$

THEOREM 6: *First-visit MC is unbiased, i.e.,* $Bias_n^F(s) = V(s) - E\{V_n^F(s)\} = 0$ *for all* $n > 0$.

Proof: The first-visit MC estimate is unbiased because the total reward on a sample path from the start state s to the terminal state T is by definition an unbiased estimate of the expected total reward across all such paths. Therefore, the average of the estimates from n independent sample paths is also unbiased. See Appendix A.3 for a detailed proof. ■

THEOREM 7: *Every-visit MC is biased and, after n trials, its bias is*

$$Bias_n^E(s) = V(s) - E\{V_n^E(s)\} = \frac{2}{n+1} Bias_1^E(s) = \frac{2}{(n+1)} \left[\frac{P_s}{2P_T} R_s \right].$$

Proof: See Appendix A.4. ■

One way of understanding the bias in the every-visit MC estimate is to note that this method averages many returns for each trial. Returns from the same trial share many of the same rewards and are thus not independent. The bias becomes smaller as more trials are observed because the returns from different trials *are* independent. Another way of understanding the bias is to note that the every-visit MC estimate (3) is the ratio of two random variables. In general, the expected value of such a ratio is not the ratio of the expected values of the numerator and denominator.

COROLLARY 7a: *Every-visit MC is unbiased in the limit as $n \to \infty$.*

3.5. Variance and MSE Results

THEOREM 8: *The variance of first-visit MC is*

$$Var_n^F(s) = \frac{Var_1^F(s)}{n} = \frac{1}{n} \left[Var(r_T) + \frac{P_s}{P_T} Var(r_s) + \frac{P_s}{P_T^2} R_s^2 \right].$$

Proof: See Appendix A.5. ■

Because the first-visit MC estimate is the sample average of estimates derived from independent trials, the variance goes down as $\frac{1}{n}$. The first two terms in the variance are due to the variance of the rewards, and the third term is the variance due to the random number of revisits to state s in each trial.

COROLLARY 8a: *The MSE of first-visit MC is*

$$MSE_n^F(s) = (Bias_n^F(s))^2 + Var_n^F(s) = \frac{1}{n} \left[Var(r_T) + \frac{P_s}{P_T} Var(r_s) + \frac{P_s}{P_T^2} R_s^2 \right].$$

THEOREM 9: *The variance of every-visit MC after one trial is bounded by*

$$Var(r_s)\left[\frac{P_s}{3P_T} - \frac{1}{6}\right] + Var(r_T) + \frac{1}{4}\frac{P_s}{P_T^2}R_s^2 \leq Var_1^E(s)$$

and

$$Var_1^E(s) \leq Var(r_s)\left[\frac{P_s}{3P_T}\right] + Var(r_T) + \frac{1}{4}\frac{P_s}{P_T^2}R_s^2.$$

Proof: See Appendix A.6. ■

We were able to obtain only these upper and lower bounds on the variance of every-visit MC. For a single trial, every-visit MC produces an estimate that is closer to zero than the estimate produced by first-visit MC; therefore $Var_1^E \leq Var_1^F$. This effect was seen in the simple example of Figure 2, in which the every visit MC estimator underestimated the expected number of revisits.

Of course, a low variance is not of itself a virtue. For example, an estimator that returns a constant independent of the data has zero variance, but is not a good estimator. Of greater importance is to be low in mean squared error (MSE):

COROLLARY 9a: *After one trial, $MSE_1^E(s) \leq MSE_1^F(s)$ because $(Bias_1^E(s))^2 + Var_1^E(s) \leq MSE_1^F(s)$.*

Thus, after one trial, every-visit MC is always as good or better than first-visit MC in terms of both variance and MSE. Eventually, however, this relative advantage always reverses itself:

THEOREM 10: *There exists an $N < \infty$, such that for all $n \geq N$, $Var_n^E(s) \geq Var_n^F(s)$.*

Proof: The basic idea of the proof is that the $O(\frac{1}{n})$ component of Var_n^E is larger than that of Var_n^F. The other $O\left(\frac{1}{n^{\frac{3}{2}}}\right)$ components of Var_n^E fall off much more rapidly than the $O\left(\frac{1}{n}\right)$ component, and can be ignored for large enough n. See Appendix A.7 for a complete proof. ■

COROLLARY 10a: *There exists an $N < \infty$, such that, for all $n > N$,*

$$MSE_n^E(s) = (Bias_n^E(s))^2 + Var_n^E(s) \geq MSE_n^F(s) = Var_n^F(s).$$

Figure 4 shows an empirical example of this *crossover* of MSE. These data are for the two MC methods applied to an instance of the example task of Figure 2a. In this case crossover occurred at trial $N = 5$. In general, crossover can occur as early as the first trial. For example, if the only non-zero reward in a problem is on termination, then $R_s = 0$, and $Var(r_s) = 0$, which in turn implies that $Bias_n^E = 0$, for all n, and that $Var_1^E(s) = Var_1^F(s)$, so that $MSE_1^E(s) = MSE_1^F(s)$.

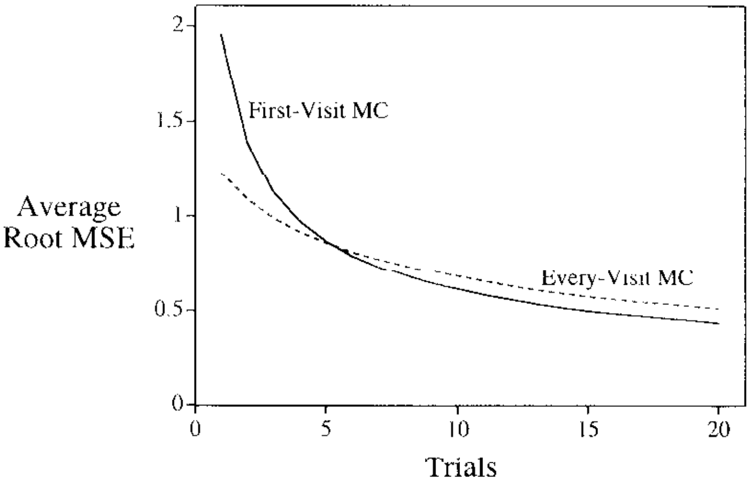

Figure 4. Empirical demonstration of crossover of MSE on the example task shown in Figure 2a. The S-to-S transition probability was $p = 0.6$. These data are averages over $10,000$ runs.

3.6. *Summary*

Table 1 summarizes the results of this section comparing first-visit and every-visit MC methods. Some of the results are unambiguously in favor of the first-visit method over the every visit method: only the first visit estimate is unbiased and related to the ML estimate. On the other hand, the MSE results can be viewed as mixed. Initially, every-visit MC is of better MSE, but later it is always overtaken by first-visit MC. The implications of this are unclear. To some it might suggest that we should seek a combination of the two estimators that is always of lowest MSE. However, that might be a mistake. We suspect that the first-visit estimate is always the more useful one, even when it is worse in terms of MSE. Our other theoretical results are consistent with this view, but it remains a speculation and a topic for future research.

Table 1. Summary of Statistical Results

Algorithm	Convergent	Unbiased	Short MSE	Long MSE	Reduced-ML
First-Visit MC	Yes	Yes	Higher	Lower	Yes
Every-Visit MC	Yes	No	Lower	Higher	No

4. Random-Walk Experiment

In this section we present an empirical comparison of replacing and accumulating eligibility traces. Whereas our theoretical results are limited to the case of $\lambda = 1$ and either offline or batch updating, in this experiment we used online updating and general λ. We used the random-walk process shown in Figure 5. The rewards were zero everywhere except upon entering the terminal states. The reward upon transition into State 21 was $+1$ and upon transition into State 1 was -1. The discount factor was $\gamma = 1$. The initial value estimates were 0 for all states. We implemented online TD(λ) with both kinds of traces for ten different values of λ: 0.0, 0.2, 0.4, 0.6, 0.8, 0.9, 0.95, 0.975, 0.99, and 1.0.

The step-size parameter was held constant, $\alpha_t(s) = \alpha$, $\forall t$, $\forall s$. For each value of λ, we used α values between 0 and 1.0 in increments of 0.01. Each (λ, α) pair was treated as a separate algorithm, each of which we ran for 10 trials. The performance measure for a trial was the root mean squared error (RMSE) between the correct predictions and the predictions made at the end of the trial from states that had been visited at least once in that or a previous trial. These errors were then averaged over the 10 trials, and then over 1000 separate runs to obtain the performance measure for each algorithm plotted in Figures 6 and 7. The random number generator was seeded such that all algorithms experienced exactly the same trials.

Figure 6 shows the performance of each method as a function of α and λ. For each value of λ, both kinds of TD method performed best at an intermediate value of α, as is typically the case for such learning algorithms. The larger the λ value, the smaller the α value that yielded best performance, presumably because the eligibility trace multiplies the step-size parameter in the update equation.

The critical results are the differences between replace and accumulate TD methods. Replace TD was much more robust to the choice of the step-size parameter than accumulate TD. Indeed, for $\lambda \geq 0.9$, accumulate TD(λ) became unstable for $\alpha > 0.6$. At large λ, accumulate TD built up very large eligibility traces for states that were revisited many times before termination. This caused very large changes in the value estimates and led to instability. Figure 7 summarizes the data by plotting, for each λ, only the performance at the best α for that λ. For every λ, the best performance of replace TD was better than or equal to the best performance of accumulate TD. We conclude that, at least for the problem studied here, replace TD(λ) is faster and more robust than accumulate TD(λ).

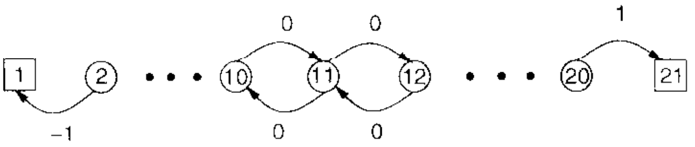

Figure 5. The random-walk process. Starting in State 11, steps are taken left or right with equal probability until either State 1 or State 21 is entered, terminating the trial and generating a final non-zero reward.

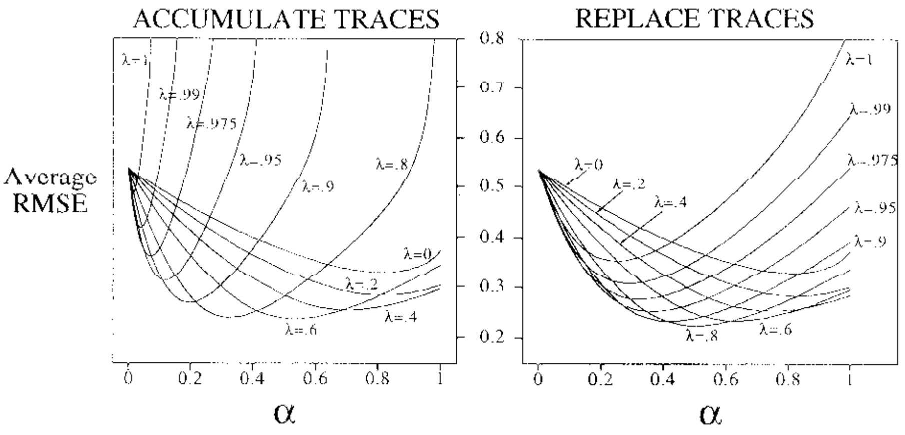

Figure 6. Performance of replace and accumulate TD(λ) on the random-walk task, for various values of λ and α. The performance measure was the RMSE per state per trial over the first 10 trials. These data are averages over 1000 runs.

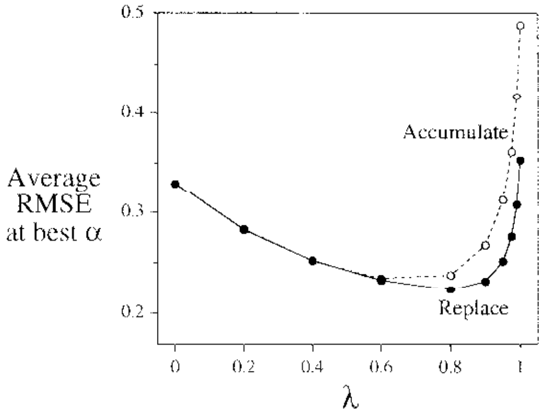

Figure 7. Best performances of accumulate and replace TD(λ) on the random-walk task.

5. Mountain-Car Experiment

In this section we describe an experimental comparison of replacing and accumulating traces when used as part of a reinforcement learning system to solve a control problem. In this case, the methods learned to predict the value not of a state, but of a state-action pair, and the approximate value function was implemented as a set of CMAC neural networks, one for each action.

The control problem we used was Moore's (1991) *mountain car* task. A car drives along a mountain track as shown in Figure 8. The objective is to drive past the top of the mountain on the righthand side. However, gravity is stronger than the engine, and even at full thrust the car cannot accelerate up the steep slope. The only way to solve the problem is to first accelerate *backwards*, away from the goal, and then apply full thrust forwards, building up enough speed to carry over the steep slope even while slowing down the whole way. Thus, one must initially move away from the goal in order to reach it in the long run. This is a simple example of a task where things must get worse before they can get better. Many control methodologies have great difficulties with tasks of this kind unless explicitly aided by a human designer.

The reward in this problem is -1 for all time steps until the car has passed to the right of the mountain top. Passing the top ends the trial and ends this punishment. The reinforcement learning agent seeks to maximize its total reward prior to the termination of the trial. To do so, it must drive to the goal in minimum time. At each time step the learning agent chooses one of three actions: full thrust forward, full thrust reverse, or no thrust. This action, together with the effect of gravity (dependent on the steepness of the slope), determines the next velocity and position of the car. The complete physics of the mountain-car task are given in Appendix B.

The reinforcement learning algorithm we applied to this task was the *Sarsa* algorithm studied by Rummery and Niranjan (1994) and others. The objective in this algorithm is to learn to estimate the action-value function $Q^\pi(s, a)$ for the current policy π. The action-

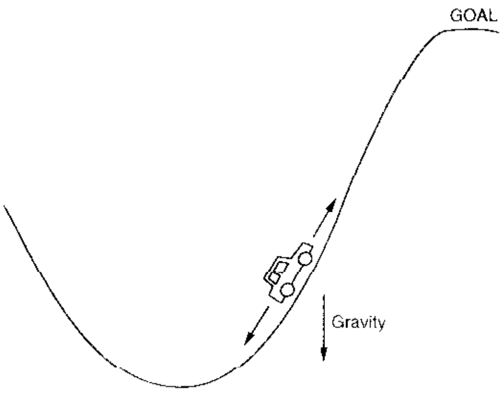

Figure 8. The Mountain-Car task. The force of gravity is stronger than the motor.

```
1. Initially: $w_a(f) := -20$, $e_a(f) := 0$, $\forall a \in \mathit{Actions}$, $\forall f \in$ CMAC-tiles.

2. Start of Trial: $s :=$ random-state();
                   $F := \mathit{features}(s)$;
                   $a :=$ greedy-policy$(F)$.

3. Eligibility Traces: $e_b(f) := \lambda e_b(f)$, $\forall b$, $\forall f$;
      3a. Accumulate algorithm: $e_a(f) := e_a(f) + 1$, $\forall f \in F$.
      3b. Replace algorithm:    $e_a(f) := 1$, $e_b(f) := 0$, $\forall f \in F$, $\forall b \neq a$.

4. Environment Step:
      Take action $a$; observe resultant reward, $r$, and next state $s'$.

5. Choose Next Action:
      $F' := \mathit{features}(s')$, unless $s'$ is the terminal state, then $F' := \emptyset$;
      $a' :=$ greedy-policy$(F')$.

6. Learn: $w_b(f) := w_b(f) + \frac{\alpha}{m}[r + \sum_{f \in F'} w_{a'} - \sum_{f \in F} w_a] e_b(f)$, $\forall b, \forall f$.

7. Loop: $a := a'$; $s := s'$; $F := F'$; if $s'$ is the terminal state, go to 2; else go to 3.
```

Figure 9. The Sarsa Algorithm used on the Mountain-Car task. The function *greedy-policy*(F) computes $\sum_{f \in F} w_a$ for each action a and returns the action for which the sum is largest, resolving any ties randomly. The function *features*(s) returns the set of CMAC tiles corresponding to the state s. Programming optimizations can reduce the expense per iteration to a small multiple (dependent on λ) of the number of features, m, present on a typical time step. Here m is 5.

value $Q^{\pi}(s,a)$ gives, for any state, s, and action, a, the expected return for starting from state s, taking action a, and thereafter following policy π. In the case of the mountain-car task the return is simply the sum of the future reward, i.e., the negative of the number of time steps until the goal is reached. Most of the details of the Sarsa algorithm we used are given in Figure 9. The name "Sarsa" comes from the quintuple of actual events involved in the update: $(s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1})$. This algorithm is closely related to Q-learning (Watkins, 1989) and to various simplified forms of the bucket brigade (Holland, 1986; Wilson, to appear). It is also identical to the TD(λ) algorithm applied to state-action pairs rather than to states.[6]

The mountain-car task has a continuous two-dimensional state space with an infinite number of states. To apply reinforcement learning requires some form of function approximator. We used a set of three CMACs (Albus, 1981; Miller, Glanz, & Kraft, 1990), one for each action. These are simple functions approximators using repeated overlapping tilings of the state space to produce a feature representation for a final linear mapping. In this case we divided the two state variables, the position and velocity of the car, each into eight evenly spaced intervals, thereby partitioning the state space into 64 regions, or boxes. A ninth row and column were added so that the tiling could be offset by a random fraction of an interval without leaving any states uncovered. We repeated this five times, each with a different, randomly selected offset. For example, Figure 10 shows two tilings superimposed on the 2D state space. The result was a total of $9 \times 9 \times 5 = 405$ boxes. The state at any particular time was represented by the five boxes, one per tiling, within which the state resided. We think of the state representation as a feature vector with 405 features, exactly 5 of which are present (non-zero) at any point in time. The approximate action-value function is linear in this feature representation. Note that this representation of the state causes the problem to no longer be Markov: many different nearby states produce exactly the same feature representation.

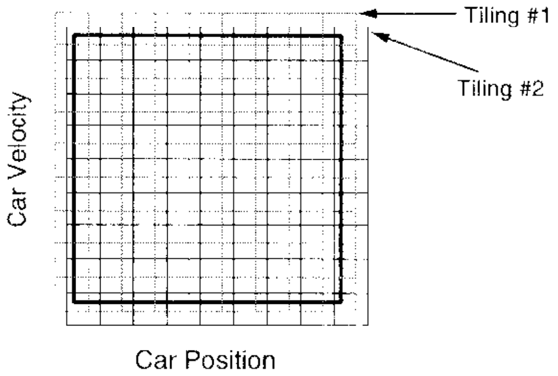

Figure 10. Two 9×9 CMAC tilings offset and overlaid over the continuous, two-dimensional state space of the Mountain-Car task. Any state is in exactly one tile/box/feature of each tiling. The experiments used 5 tilings, each offset by a random fraction of a tile width.

The eligibility traces were implemented on a feature-by-feature basis. Corresponding to each feature were three traces, one per action. The features are treated in essence like states. For replace algorithms, whenever a feature occurs, its traces are reset to 1 (for the action selected) or 0 (for all the other actions). This is not the only possibility, of course. Another would be to allow the traces for each state-action pair to continue until that *pair* occurred again. This would be more in keeping with the idea of replacing traces as a mechanism, but the approach we chose seems like the appropriate way to generalize the idea of first-visit MC to the control case: after a state has been revisited, it no longer matters what action was taken on the previous visit. A comparison of these two possibilities (and perhaps others) would make a good extension to this work.

The greedy policy was used to select actions. The initial weights were set to produce a uniform, optimistic initial estimate of value (-100) across the state space.[7] See Figure 9 for further details.

We applied replace and accumulate Sarsa algorithms to this task, each with a range of values for λ and α. Each algorithm was run for 20 trials, where a trial was one passage from a randomly selected starting state to the goal. All algorithms used the same sets of random starting states. The performance measure for the run was the average trial length over the 20 trials. This measure was then averaged over 30 runs to produce the results shown in Figures 11 and 12. Figure 11 shows the detailed results for each value of λ and α, whereas Figure 12 is a summary showing only the best performance of each algorithm at each λ value.

Several interesting results are evident from this data. First, the replace-trace method performed better than the accumulate-trace method at all λ values. The accumulate method performed particularly poorly relative to the replace method at high values of λ. For both methods, performance appeared to be best at an intermediate λ value. These

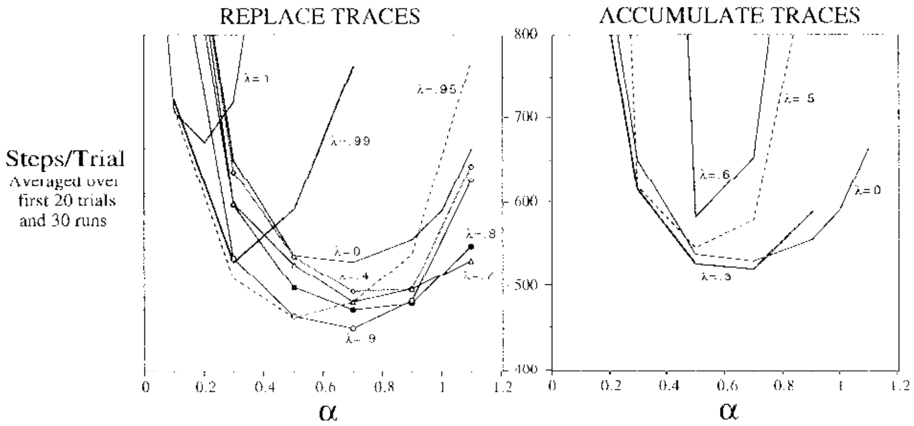

Figure 11. Results on the Mountain-Car task for each value of λ and α. Each data point is the average duration of the first 20 trials of a run, averaged over 30 runs. The standard errors are omitted to simplify the graph; they ranged from about 10 to about 50.

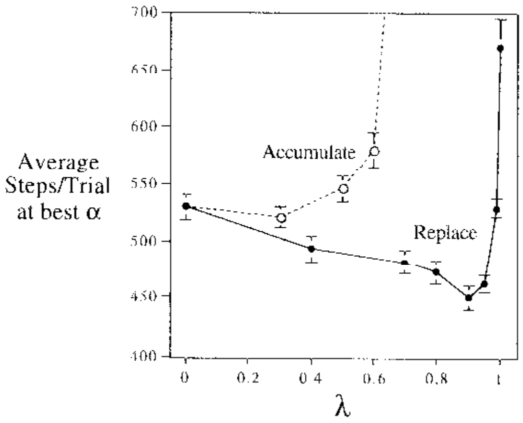

Figure 12. Summary of results on the Mountain-Car task. For each value of λ we show its performance at its best value of α. The error bars indicate one standard error.

results are all consistent with those presented for the random-walk task in the previous section. On the mountain-car task, accumulating traces at best improved only slightly over no traces ($\lambda = 0$) and at worst dramatically degraded performance. Replacing traces, on the other hand, significantly improved performance at all except the very longest trace lengths ($\lambda > .99$). Traces that do not decay ($\lambda = 1$) resulted in significantly worse performance than all other λ values tried, including no traces at all ($\lambda = 0$).

Much more empirical experience is needed with trace mechanisms before a definitive conclusion can be drawn about their relative effectiveness, particularly when function approximators are used. However, these experiments do provide significant evidence for two key points: 1) that replace-trace methods can perform much better than conventional, accumulate-trace methods, particularly at long trace lengths, and 2) that although long traces may help substantially, best performance is obtained when the traces are not infinite, that is, when intermediate predictions are used as targets rather than actual sample returns.

6. Conclusions

We have presented a variety of analytical and empirical evidence supporting the idea that replacing eligibility traces permit more efficient use of experience in reinforcement learning and long-term prediction.

Our analytical results concerned a special case closely related to that used in classical studies of Monte Carlo methods. We showed that methods using conventional traces are biased, whereas replace-trace methods are unbiased. While the conclusions of our mean-squared-error analysis are mixed, the maximum likelihood analysis is clearly in favor of replacing traces. As a whole, these analytic results strongly support the conclusion that replace-trace methods make better inferences from limited data than conventional accumulate-trace methods.

On the other hand, these analytic results concern only a special case quite different from those encountered in practice. It would be desirable to extend our analyses to the case of $\lambda < 1$ and to permit other step-size schedules. Analysis of cases involving function approximators and violations of the Markov assumption would also be useful further steps.

Our empirical results treated a much more realistic case, including in some cases all of the extensions listed above. These results showed consistent, significant, and sometimes large advantages of replace-trace methods over accumulate-trace methods, and of trace methods generally over trace-less methods. The mountain-car experiment showed that the replace-trace idea can be successfully used in conjunction with a feature-based function approximator. Although it is not yet clear how to extend the replace-trace idea to other kinds of function approximators, such as back-propagation networks or nearest-neighbor methods, Sutton and Whitehead (1993) and others have argued that feature-based function approximators are actually preferable for online reinforcement learning.

Our empirical results showed a sharp drop in performance as the trace parameter λ approached 1, corresponding to very long traces. This drop was much less severe with replacing traces but was still clearly present. This bears on the long-standing question of the relative merits of TD(1) methods versus true temporal-difference ($\lambda < 1$) methods. It might appear that replacing traces make TD(1) methods more capable competitors; the replace TD(1) method is unbiased in the special case, and more efficient than conventional TD(1) in both theory and practice. However, this is at the cost of losing some of the theoretical advantages of conventional TD(1). In particular, conventional TD(1) converges in many cases to a minimal mean squared-error solution when function approximators are used (Dayan, 1992) and has been shown to be useful in non-Markov problems (Jaakkola, Singh & Jordan, 1995). The replace version of TD(1) does not share these theoretical guarantees. Like $\lambda < 1$ methods, it appears to achieve greater efficiency in part by relying on the Markov property. In practice, however, the relative merits of different $\lambda = 1$ methods may not be of great significance. All of our empirical results suggest far better performance is obtained with $\lambda < 1$, even when function approximators are used that create an apparently non-Markov task.

Replacing traces are a simple modification of existing discrete-state or feature-based reinforcement learning algorithms. In cases in which a good state representation can be obtained they appear to offer significant improvements in learning speed and reliability.

Acknowledgments

We thank Peter Dayan for pointing out and helping to correct errors in the proofs. His patient and thorough reading of the paper and his participation in our attempts to complete the proofs were invaluable. Of course, any remaining errors are our own. We thank Tommi Jaakkola for providing the central idea behind the proof of Theorem 10, an anonymous reviewer for pointing out an error in our original proof of Theorem 10, and Lisa White for pointing out another error. Satinder P. Singh was supported by grants to

Michael I. Jordan (Brain and Cognitive Sciences, MIT) from ATR Human Information Processing Research and from Siemens Corporation.

Appendix A

Proofs of Analytical Results

A.1. Proof of Theorem 5: First-Visit MC is Reduced-ML

In considering the estimate $V_t(s)$, we can assume that all trials start in s, because both first-visit MC and reduced-ML methods ignore transitions prior to the first visit to s. Let n_i be the number of times state i has been visited, and let n_{ij} be the number of times transition $i \rightsquigarrow j$ has been encountered. Let R_{jk} be the average of the rewards seen on the $j \rightsquigarrow k$ transitions.

Then $V_N^F(s)$, the first-visit MC estimate after N trials with start state s, is

$$V_N^F(s) = \frac{1}{N} \sum_{k\in S, j\in S} n_{jk} R_{jk}.$$

This is identical to (2) because $\sum_{k\in S, j\in S} n_{jk} R_{jk}$ is the total summed reward seen during the N trials. Because $N = n_s - \sum_{i\in S} n_{is}$, we can rewrite this as

$$V_N^F(s) = \sum_{k\in S, j\in S} \frac{n_{jk}}{n_s - \sum_{i\in S} n_{is}} R_{jk} = \sum_{j,k} U'_{jk} R_{jk}. \tag{A.1}$$

The maximum-likelihood model of the Markov process after N trials has transition probabilities $P(ij) = \frac{n_{ij}}{n_i}$ and expected rewards $R(ij) = R_{ij}$. Let $V_N^{ML}(s)$ denote the reduced-ML estimate after N trials. By definition $V_N^{ML}(s) = E_N\{r_1+r_2+r_3+r_4+\ldots\}$, where E_N is the expectation operator for the maximum-likelihood model after N trials, and r_i is the payoff at step i. Therefore

$$\begin{aligned} V_N^{ML}(s) &= \sum_{j,k} R_{jk} \left[Prob_1(j \rightsquigarrow k) + Prob_2(j \rightsquigarrow k) + Prob_3(j \rightsquigarrow k) + \ldots\right] \\ &= \sum_{j,k} R_{jk} U_{jk}, \end{aligned} \tag{A.2}$$

where $Prob_i(j \rightsquigarrow k)$ is the probability of a j-to-k transition at the i^{th} step according to the maximum-likelihood model. We now show that for all j, k, U_{jk} of (A.2) is equal to U'_{jk} of (A.1).

Consider two special cases of j in (A.2):

Case 1, $j = s$:

$$\begin{aligned} U_{sk} &= P(sk) + P(ss)P(sk) + \sum_m P(sm)P(ms)P(sk) + \ldots \\ &= \frac{n_{sk}}{n_s}\left[1 + \frac{n_{ss}}{n_s} + \sum_m \frac{n_{sm}}{n_s}\frac{n_{ms}}{n_m} + \ldots\right] \end{aligned}$$

$$\begin{aligned}
&= \frac{n_{sk}}{n_s}\left[1 + P^1(ss) + P^2(ss) + P^3(ss) + \ldots\right] \\
&= \frac{n_{sk}}{n_s}(1 + N_{ss}),
\end{aligned} \tag{A.3}$$

where $P^n(ij)$ is the probability of going from state i to state j in exactly n steps, and N_{ss} is the expected number of revisits to state s as per our current maximum-likelihood model.

Case 2, $j \neq s$:

$$\begin{aligned}
U_{jk} &= P(sj)P(jk) + \sum_m P(sm)P(mj)P(jk) + \ldots \\
&= \frac{n_{jk}}{n_j}\left[\frac{n_{sj}}{n_s} + \sum_m \frac{n_{sm}}{n_s}\frac{n_{mj}}{n_m} + \ldots\right] \\
&= \frac{n_{jk}}{n_j}\left[P^1(sj) + P^2(sj) + P^3(sj) + \ldots\right] \\
&= \frac{n_{jk}}{n_j}N_{sj},
\end{aligned} \tag{A.4}$$

where N_{sj} is the expected number of visits to state j.

For all j, the N_{sj} satisfy the recursions

$$N_{sj} = P(sj) + \sum_m N_{sm}P(mj) = \frac{n_{sj}}{n_s} + \sum_m N_{sm}\frac{n_{mj}}{n_m}. \tag{A.5}$$

We now show that $N_{sj} = \frac{n_j}{n_s - \sum_i n_{is}}$ for $j \neq s$, and $N_{ss} = \frac{n_s}{n_s - \sum_i n_{is}} - 1$, by showing that these quantities satisfy the recursions (A.5).

$$\begin{aligned}
N_{sj} &= \frac{n_{sj}}{n_s} + \sum_{m \neq s}\left(\frac{n_{mj}}{n_m}\frac{n_m}{n_s - \sum_i n_{is}}\right) + \frac{n_{sj}}{n_s}\left(\frac{n_s}{n_s - \sum_i n_{is}} - 1\right) \\
&= \frac{n_{sj}}{n_s} + \sum_m \frac{n_{mj}}{n_s - \sum_i n_{is}} - \frac{n_{sj}}{n_s} \\
&= \frac{\sum_m n_{mj}}{n_s - \sum_i n_{is}} = \frac{n_j}{n_s - \sum_i n_{is}},
\end{aligned}$$

and, again from (A.5),

$$\begin{aligned}
N_{ss} &= \frac{n_{ss}}{n_s} + \sum_{m \neq s}\left(\frac{n_{ms}}{n_m}\frac{n_m}{n_s - \sum_i n_{is}}\right) + \frac{n_{ss}}{n_s}\left(\frac{n_s}{n_s - \sum_i n_{is}} - 1\right) \\
&= \frac{\sum_m n_{ms}}{n_s - \sum_i n_{is}} = \frac{n_s}{n_s - \sum_i n_{is}} - 1.
\end{aligned}$$

Plugging the above values of N_{ss} and N_{sj} into (A.3) and (A.4), we obtain $U_{jk} = \frac{n_{jk}}{n_s - \sum_i n_{is}} = U'_{jk}$. ■

A.2. Facts Used in Proofs of Theorems 6-10

The proofs for Theorems 6-10 assume the abstract chain of Figure 3 with just two states, s and T. The quantities $R_s = E\{r_s\}$, $Var(r_s) = E\{(r_s - R_s)^2\}$, $R_T = E\{r_T\}$, and $Var(r_T) = E\{(r_T - R_T)^2\}$ are of interest for the analysis and require careful elaboration. Let S_s be the set of all state sequences that can occur between visits to state s (including state s at the head), and let S_T be the set of all state sequences that can occur on the final run from s to T (including state s). The termination probability is $P_T = \sum_{\{s\}_T \in S_T} P(\{s\}_T)$, where the probability of a sequence of states is the product of the probabilities of the individual state transitions. By definition $P_s = 1 - P_T$. The reward probabilities are defined as follows: $Prob\{r_s = q\} = \Pi_{\{s\} \in S_s} P(\{s\}) P(r_{\{s\}} = q | \{s\})$, and $Prob\{r_T = q\} = \Pi_{\{s\}_T \in S_T} P(\{s\}_T) P(r_T = q | \{s\}_T)$. Therefore, $R_s = \sum_q q Prob\{r_s = q\}$, $R_T = \sum_q q Prob\{r_T = q\}$. Similarly, $Var(r_s) = \sum_q Prob\{r_s = q\}(q - R_s)^2$, and $Var(r_T) = \sum_q Prob\{r_T = q\}(q - R_T)^2$.

If the rewards in the original Markov chain are deterministic functions of the state transitions, then there will be a single r_i associated with each $\{s\}_i$. If the rewards in the original problem are stochastic, however, then there is a set of possible random r_i's associated with each $\{s\}_i$. Also note that even if all the individual rewards in the original Markov chain are deterministic, $Var(r_s)$ and $Var(r_T)$ can still be greater than zero because r_s and r_T will be stochastic because of the many different paths from s to s and from s to T.

The following fact is used throughout:

$$\begin{aligned} E[f(\{x\})] &= \sum_{\{x\}} P(\{x\}) f(\{x\}) \\ &= \sum_k P(k) E_{\{r\}}\{f(\{x\})|k\}, \end{aligned} \tag{A.6}$$

where k is the number of revisits to state s. We also use the facts that, if $r < 1$, then,

$$\sum_{i=0}^{\infty} i r^i = \frac{r}{(1-r)^2} \quad \text{and} \quad \sum_{i=0}^{\infty} i^2 r^i = \frac{r(1+r)}{(1-r)^3}.$$

A.3. Proof of Theorem 6: First-Visit MC is Unbiased

First we show that first-visit MC is unbiased for one trial. From (A.6)

$$\begin{aligned} E\{V_1^F(s)\} = E_{\{x\}}[f(\{x\})] &= \sum_k P(k) E_{\{r\}}\{f(\{x\})|k\} \\ &= \sum_k P_T P_s^k (k R_s + R_T) \\ &= P_T \left[\frac{P_s}{(1-P_s)^2} R_s + \frac{1}{1-P_s} R_T \right] \end{aligned}$$

$$
\begin{aligned}
&= \frac{P_s}{P_T} R_s + R_T \\
&= V(s).
\end{aligned}
$$

Because the estimate after n trials, $V_n^F(s)$, is the sample average of n independent estimates each of which is unbiased, the n-trial estimate itself is unbiased. ■

A.4. Proof of Theorem 7: Every-Visit MC is Biased

For a single trial, the bias of the every-visit MC algorithm is

$$
\begin{aligned}
E\{V_1^E(s)\} - E_{\{x\}}[t(\{x\})] &= \sum_k P(k) E_{\{r\}}\{t(\{x\})|k\} \\
&= \sum_k P_T P_s^k \left(\frac{R_s + 2R_s + \ldots + kR_s + (k+1)R_T}{k+1} \right) \\
&= \sum_k P_T P_s^k \left(\frac{k}{2} R_s + R_T \right) \\
&= \frac{P_s}{2P_T} R_s + R_T.
\end{aligned}
$$

Therefore, $Bias_1^E = V(s) - E\{V_1^E(s)\} - \frac{P_s}{2P_T} R_s$.

Computing the bias after n trials is a bit more complex, because of the combinatorics of getting k revisits to state s in n trials, denoted $B(n;k)$. Equivalently, one can think of $B(n;k)$ as the number of different ways of factoring k into n non-negative integer additive factors *with the order considered important*. Therefore, $B(n;k) = \binom{k+n-1}{n-1}$. Further let $B(n;k_1,k_2,\ldots,k_n|k)$ be the number of different ways one can get k_1 to k_n as the factors *with the order ignored*. Note that $\sum_{\text{factors of } k} B(n;k_1,k_2,\ldots,k_n|k) = B(n;k)$. We use superscripts to distinguish the rewards from different trials, e.g., r_j^2 refers to the random total reward received between the j^{th} and $(j+1)^{st}$ visits to start state s in the second trial.

$$
\begin{aligned}
E\{V_n^E(s)\} &= E_{\{x\}} \left\{ \frac{\sum_{i=1}^n t_{num}(\{x\}^i)}{\sum_{i=1}^n (k_i+1)} \right\} \\
&= \sum_{k_1,k_2,\ldots,k_n} P(k_1,k_2,\ldots,k_n) E_{\{r\}} \left\{ \frac{\sum_{i=1}^n \sum_{j=1}^{k_i} j r_{s_j}^i}{\sum_{i=1}^n (k_i+1)} \middle| k_1,k_2,\ldots,k_n \right\} \\
&= \sum_k P(k) \sum_{\text{factors of } k} B(n;k_1,k_2,\ldots,k_n|k) \left[\frac{\sum_{i=1}^n k_i(k_i+1)}{2(k+n)} R_s + R_T \right] \\
&= R_T + R_s \left(\sum_k P_T^n P_s^k \sum_{\text{factors of } k} B(n;k_1,k_2,\ldots,k_n|k) \left[\frac{\sum_{i=1}^n (k_i)^2 + k}{2(k+n)} \right] \right).
\end{aligned}
$$

This, together with

$$\sum_{\text{factors of } k} B(n, k_1, k_2, \ldots, k_n|k)\left[\frac{\sum_{i=1}^{n}(k_i)^2 + k}{2(k+n)}\right] = \frac{k}{n+1}B(n;k), \tag{A.7}$$

which we show below, and the fact that

$$\sum_{k=0}^{\infty} P_s^k P_T^n B(n;k)\frac{k}{n} = \frac{P_s}{P_T},$$

leads to the conclusion that

$$E\{V_n^E(s)\} = R_T + \frac{n}{n+1}\frac{P_s}{P_T}R_s.$$

Therefore $Bias_n^E(s) = V(s) - E\{V_n^E(s)\} = \frac{1}{n+1}\frac{P_s}{P_T}R_s$. This also proves that the every-visit MC algorithm is unbiased in the limit as $n \rightarrow \infty$. ■

Proof of (A.7):

Define $T(n;k)$ as the sum over all factorizations of the squared factors of k:

$$T(n;k) = \sum_{\text{factors of } k} B(n;k_1,k_2,\ldots,k_n|k)\sum_{i=1}^{n}(k_i)^2.$$

We know the following facts from first principles:

$$\text{Fact 1: } \sum_{j=0}^{k} B(n;k-j) = B(n+1;k);$$

$$\text{Fact 2: } \sum_{j=0}^{k} jB(n;k-j) = \frac{kB(n+1;k)}{n+1};$$

$$\text{Fact 3: } \sum_{j=0}^{k} j^2B(n;k-j) = \frac{T(n+1;k)}{n+1};$$

and also that, by definition,

$$\text{Fact 4: } T(n+1;k) = \sum_{j=0}^{k} T(n;k-j) - j^2B(n;k-j).$$

Facts 3 and 4 imply that

$$T(n+1;k) = \frac{n+1}{n}\sum_{j=0}^{k} T(n;k-j). \tag{A.8}$$

Using Facts 1-3 we can show by substitution that $T(n;k) = \frac{k(2k+n-1)}{n+1}B(n;k)$ is a solution of the recursion (A.8), hence proving that

$$\sum_{\text{factors of } k} B(n;k_1,k_2,\ldots,k_n|k)\left[\sum_{i=1}^{n}(k_i)^2\right] = \frac{k(2k+n-1)}{n+1}B(n;k) \Rightarrow$$

$$\sum_{\text{factors of } k} B(n;k_1,k_2,\ldots,k_n|k)\left[\frac{\sum_{i=1}^{n}(k_i)^2}{2(k+n)}\right] + \frac{k}{2(k+n)}B(n;k) = \frac{k}{n+1}B(n;k) \Rightarrow$$

$$\sum_{\text{factors of } k} B(n;k_1,k_2,\ldots,k_n|k)\left[\frac{\sum_{i=1}^{n}(k_i)^2+k}{2(k+n)}\right] = \frac{k}{n+1}B(n;k). \qquad \blacksquare$$

A.5. Proof of Theorem 8: Variance of First-Visit MC

We first compute the variance of first-visit MC after one trial $\{x\}$:

$$\begin{aligned}
Var_1^F(s) &= E_{\{x\}}\{(f(\{x\})-V(s))^2\} \\
&= \sum_k P(k)E_{\{r\}}\{(f(\{x\})-V(s))^2|k\} \\
&= \sum_k P_T P_s^k E_{\{r\}}\left\{\left((\sum_{i=1}^{k} r_{s_i}+r_T)-(\frac{P_s}{P_T}R_s+R_T)\right)^2 \middle| k\right\} \\
&= \sum_k P_T P_s^k E_{\{r\}}\left\{\sum_{i=1}^{k}(r_{s_i}^2+2r_T r_{s_i})+2\sum_{i\neq j} r_{s_i}r_{s_j}+r_T^2+\frac{P_s^2}{P_T^2}R_s^2\right. \\
&\quad \left.+2\frac{P_s}{P_T}R_sR_T+R_T^2-\sum_{i=1}^{k}2r_{s_i}\left(\frac{P_s}{P_T}R_s+R_T\right)-2r_T\left(\frac{P_s}{P_T}R_s+R_T\right)\middle| k\right\} \\
&= \sum_k P_T P_s^k\left[E_{\{r\}}\left\{\sum_{i=1}^{k}(r_{s_i}^2-R_s^2)+(r_T^2-R_T^2)\middle| k\right\}+k^2R_s^2+2kR_sR_T\right. \\
&\quad \left.+\frac{P_s^2}{P_T^2}R_s^2+R_T^2+2\frac{P_s}{P_T}R_sR_T-2k\frac{P_s}{P_T}R_s^2-2\frac{P_s}{P_T}R_sR_T-R_T^2-2kR_sR_T\right] \\
&= \sum_k P_T P_s^k\left[kVar(r_s)+Var(r_T)+R_s^2\left(k-\frac{P_s}{P_T}\right)^2\right] \\
&= \frac{P_s}{P_T^2}R_s^2+Var(r_T)+\frac{P_s}{P_T}Var(r_s).
\end{aligned}$$

The first term is the variance due to the variance in the number of revisits to state s, and the second term is the variance due to the random rewards. The first-visit MC estimate after n trials is the sample average of the n independent trials; therefore, $Var_n^F(s) = \frac{Var_1^F(s)}{n}$. $\blacksquare$

A.6. Proof of Theorem 9: Variance of Every-Visit MC After 1 Trial

$$\begin{aligned}
Var_1^E(s) &= E_{\{x\}}\{(t(\{x\}) - E\{t(\{x\})\})^2\} \\
&= \sum_{\{x\}} P(\{x\}) \left[\frac{r_{s_1} + 2r_{s_2} + \ldots + kr_{s_k} + (k+1)r_T}{k+1} - \left(\frac{P_s}{2P_T}R_s + R_T\right)\right]^2 \\
&= \sum_k P(k) E_{\{r\}} \left\{ \sum_{i=1}^k \left(\frac{i^2}{(k+1)^2} r_{s_i}^2 + 2\frac{i}{k+1} r_{s_i} r_T\right) + r_T^2 \right. \\
&\qquad + \sum_{i \neq j} \frac{2ij}{(k+1)^2} r_{s_i} r_{s_j} + \frac{P_s^2}{4P_T^2} R_s^2 + \frac{P_s}{P_T} R_s R_T + R_T^2 \\
&\qquad\qquad \left. -2\sum_{i=1}^k \frac{i}{k+1} r_{s_i} \left(\frac{P_s}{2P_T}R_s + R_T\right) - 2\frac{P_s}{2P_T} R_s r_T - 2R_T r_T \,\middle|\, k \right\} \\
&= \sum_k P_T P_s^k \left[Var(r_T) + \frac{k(2k+1)}{6(k+1)} Var(r_s) + \frac{R_s^2}{4}\left(k - \frac{P_s}{P_T}\right)^2 \right] \\
&= Var(r_T) + \frac{R_s^2}{4}\frac{P_s}{P_T^2} + Var(r_s) \sum_k P_T P_s^k \frac{k(2k+1)}{6(k+1)}.
\end{aligned}$$

Note that $\frac{k}{3} - \frac{1}{6} < \frac{k(2k+1)}{6(k+1)} = \frac{k}{3} - \frac{k}{6(k+1)} \leq \frac{k}{3}$. Therefore,

$$Var(r_T) + \frac{1}{4}\frac{P_s}{P_T^2} R_s^2 + Var(r_s)\left[\frac{P_s}{3P_T} - \frac{1}{6}\right] \leq Var_1^E(s),$$

and

$$Var_1^E(s) \leq Var(r_T) + \frac{1}{4}\frac{P_s}{P_T^2} R_s^2 + Var(r_s)\left[\frac{P_s}{3P_T}\right]. \qquad \blacksquare$$

A.7. Proof of Theorem 10: First-Visit MC is Eventually Lower in Variance and MSE

The central idea for this proof was provided to us by Jaakkola (personal communication). The every-visit MC estimate,

$$V_n^E(s) = \frac{\sum_{i=1}^n t_{num}(\{x\}^i)}{\sum_{i=1}^n (k_i + 1)},$$

can be rewritten as

$$V_n^E(s) = \frac{\frac{1}{n}\frac{\sum_{i=1}^n t_{num}(\{x\}^i)}{E\{(k_i+1)\}}}{\frac{1}{n}\frac{\sum_{i=1}^n (k_i+1)}{E\{k_i+1\}}} = \frac{\frac{1}{n}\sum_{i=1}^n P_T t_{num}(\{x\}^i)}{\frac{1}{n}\sum_{i=1}^n P_T(k_i + 1)}$$

because, for all i, $E\{k_i+1\} = \frac{1}{P_T}$. It is also easy to show that for all i, $E\{P_T t_{num}(\{x\}^i)\} = V(s)$, and $E\{P_T(k_i+1)\} = 1$.

Consider the sequence of functions

$$f_n(\delta) = \frac{V(s) + \delta \bar{T}_n}{1 + \delta \bar{K}_n},$$

where $\bar{T}_n = \frac{1}{\sqrt{n}} \sum_{i=1}^{n} (P_T t_{num}(\{x\}^i) - V(s))$, and $\bar{K}_n = \frac{1}{\sqrt{n}} \sum_{i=1}^{n} (P_T(k_i+1) - 1)$. Note that $\forall n$, $E\{\bar{T}_n\} = 0$, $E\{\bar{K}_n\} = 0$, and $f_n\left(\frac{1}{\sqrt{n}}\right) = V_n^E(s)$. Therefore, $Var_n^E(s) = E\{(f_n(\frac{1}{\sqrt{n}}))^2\} - (E\{V_n^E(s)\})^2$. Using Taylor's expansion,

$$f_n^2\left(\frac{1}{\sqrt{n}}\right) = f_n^2(0) + \frac{1}{\sqrt{n}} \frac{\partial}{\partial \delta} f_n^2(\delta) \Big|_{\delta=0} + \frac{1}{n2!} \frac{\partial^2}{\partial \delta^2} f_n^2(\delta) \Big|_{\delta=0} + \cdots.$$

Therefore,

$$\begin{aligned} Var_n^E(s) &= E\left\{ f_n^2(0) + \frac{1}{\sqrt{n}} \frac{\partial}{\partial \delta} f_n^2(\delta) \Big|_{\delta=0} + \frac{1}{2n} \frac{\partial^2}{\partial \delta^2} f_n^2(\delta) \Big|_{\delta=0} \right\} \\ &\quad + E\left\{ \frac{1}{6n^{\frac{3}{2}}} \frac{\partial^3}{\partial \delta^3} f_n^2(\delta) \Big|_{\delta=0} + \cdots \right\} - \left(E\{V_n^E(s)\}\right)^2. \end{aligned} \qquad (A.9)$$

We prove below that $E\left\{\frac{1}{6n^{\frac{3}{2}}} \frac{\partial^3}{\partial \delta^3} f_n^2(\delta) \Big|_{\delta=0} + \cdots\right\}$ is $O\left(\frac{1}{n^{\frac{3}{2}}}\right)$ by showing that for all $i > 2$, $E\{\frac{\partial^i}{\partial \delta^i} f_n^2(\delta)|_{\delta=0}\}$ is O(1). The $O\left(\frac{1}{n^{\frac{3}{2}}}\right)$ term in (A.9) can be ignored, because $Var_n^F(s)$ decreases as $\frac{1}{n}$, and the goal here is to show that for large N, $Var_n^E(s) > Var_n^F(s)$ for all $n > N$.

We use the following facts without proof (except 12 which is proved subsequently):

Fact 5: $E\{f_n^2(0)\} = V^2(s)$;

Fact 6: $E\left\{\frac{\partial}{\partial \delta} f_n^2(\delta) \Big|_{\delta=0}\right\} = 0$ because $\frac{\partial}{\partial \delta} f_n^2(\delta) \Big|_{\delta=0} = 2V(s)(\bar{T}_n - V(s)\bar{K}_n)$;

Fact 7: $\frac{\partial^2}{\partial \delta^2} f_n^2(\delta) \Big|_{\delta=0} = 2\bar{T}_n^2 - 8V(s)\bar{T}_n\bar{K}_n + 6V^2(s)\bar{K}_n^2$;

Fact 8: $E\{\bar{K}_n^2\} = P_s$;

Fact 9: $E\{\bar{K}_n\bar{T}_n\} = \frac{2P_s + P_s^2}{P_T} R_s + R_T(P_s+1) - V(s)$;

Fact 10: $E\{\bar{T}_n^2\} = \frac{P_s + P_s^2}{P_T} Var(r_s) + (P_s+1)Var(r_T) - V^2(s)$

$$+ (P_s + 1)R_T^2 + \frac{4P_s + 2P_s^2}{P_T} R_s R_T + \frac{P_s + 4P_s^2 + P_s^3}{P_T^2} R_s^2;$$

$$\text{Fact 11: } V^2(s) - (E\{V_n^E(s)\})^2 = \frac{(2n+1)P_s^2}{(n+1)^2 P_T^2} R_s^2 + \frac{2P_s}{(n+1)P_T} R_s R_T;$$

$$\text{Fact 12: } E\left\{\frac{1}{2n}\frac{\partial^2}{\partial\delta^2} f_n^2(\delta) \Big|_{\delta=0}\right\} = \frac{P_s + P_s^2}{nP_T} Var(r_s) + \frac{(P_s+1)}{n} Var(r_T)$$
$$+ \frac{P_s - P_s^2}{nP_T^2} R_s^2 - \frac{2P_s}{nP_T} R_s R_T.$$

Therefore, the $O(\frac{1}{n})$ behavior of the variance of the every-visit MC estimate is

$$Var_n^E(s) \approx \frac{P_s + P_s^2}{nP_T} Var(r_s) + \frac{(P_s+1)}{n} Var(r_T) + \frac{P_s}{nP_T^2} R_s^2 + \frac{n^2 - n - 1}{n(n+1)^2} \frac{P_s^2}{P_T^2} R_s^2.$$

Finally, comparing with

$$Var_n^F(s) = \frac{P_s}{nP_T} Var(r_s) + \frac{1}{n} Var(r_T) + \frac{P_s}{nP_T^2} R_s^2$$

proves Theorem 10. ∎

Ignoring higher order terms:

Note that $f_n^2(\delta)$ is of the form $\left(\frac{V(s)+\delta\bar{T}_n}{1+\delta\bar{K}_n}\right)^2$. Therefore, for all $i > 0$, the denominator of $\frac{\partial^i f_n^2(\delta)}{\partial\delta^i}$ is of the form $(1 + \delta\bar{K}_n)^j$ for some $j > 0$. On evaluating at $\delta = 0$, the denominator will always be 1, leaving in the numerator terms of the form $c\bar{T}_n^m\bar{K}_n^z$, where c is some constant and $m, z \geq 0$. For example, Fact 6 shows that $\frac{\partial}{\partial\delta} f_n^2(\delta) \mid_{\delta=0} = 2V(s)(\bar{T}_n - V(s)\bar{K}_n)$, and Fact 7 shows that $\frac{\partial^2}{\partial\delta^2} f_n^2(\delta) \mid_{\delta=0} = 2\bar{T}_n^2 - 8V(s)\bar{T}_n\bar{K}_n + 6V^2(s)\bar{K}_n^2$. We show that $E\{\bar{T}_n^m K_n^z\}$ is $O(1)$.

Both $\bar{T}_n$ and $\bar{K}_n$ are sums of n independent mean-zero random variables. $\bar{T}_n^m\bar{K}_n^z$ contains terms that are products of random variables for the n trials. On taking expectation, any term that contains a random variable from some trial exactly once drops out. Therefore all terms that remain should contain variables from no more than $\lfloor\frac{m+z}{2}\rfloor$ trials. This implies that the number of terms that survive are $O(n^{\lfloor\frac{m-z}{2}\rfloor})$ (the constant is a function of i, but is independent of n). Therefore,

$$E\{T_n^m K_n^z\} = \frac{1}{n^{\frac{m+z}{2}}} n^{\lfloor\frac{m-z}{2}\rfloor} c_1 \text{ which is } O(1),$$

where c_1 is some constant. This implies that the expected value of the terms in the Taylor expansion corresponding to $i > 2$ are $O\left(\frac{1}{n^{\frac{3}{2}}}\right)$. ∎

Proof of Fact 12:

$$E\left\{\frac{1}{2n}\frac{\partial^2}{\partial\delta^2}f_n^2(\delta)\,\Big|_{\delta=0}\right\} = \frac{E\{2\bar{T}_n^2 - 8V(s)\bar{T}_n\bar{K}_n + 6V^2(s)\bar{K}_n^2\}}{2n}. \tag{A.10}$$

From Fact 10:

$$\begin{aligned} E\{2\bar{T}_n^2\} &= 2\frac{P_s + P_s^2}{P_T}Var(r_s) + 2(P_s + 1)Var(r_T) - 2V^2(s) \\ &\quad + 2(1 + P_s)\left(\frac{P_s^2}{P_T^2}R_s^2 + 2\frac{P_s}{P_T}R_sR_T + R_T^2\right) \\ &\quad + \frac{4P_s}{P_T}R_sR_T + \frac{2P_s}{P_T^2}R_s^2 + \frac{6P_s^2}{P_T^2}R_s^2 \\ &= 2\frac{P_s + P_s^2}{P_T}Var(r_s) + 2(P_s + 1)Var(r_T) \\ &\quad + 2P_sV^2(s) + \frac{4P_s}{P_T}R_sR_T + \frac{2P_s}{P_T^2}R_s^2 + \frac{6P_s^2}{P_T^2}R_s^2. \end{aligned}$$

Similarly, from Fact 9:

$$\begin{aligned} E\{-8V(s)\bar{T}_n\bar{K}_n\} &= -8V(s)\left(\frac{P_s}{P_T}R_s + (1 + P_s)(\frac{P_s}{P_T}R_s + R_T) - V(s)\right) \\ &= -8V^2(s)P_s - \frac{8P_s^2}{P_T^2}R_s^2 - \frac{8P_s}{P_T}R_sR_T, \end{aligned}$$

and from Fact 8:

$$E\{6V^2(s)\bar{K}_n^2\} = 6P_sV_s^2.$$

Therefore, from (A.10), we see that
$E\{\frac{1}{2n}\frac{\partial^2}{\partial\delta^2}f^2(\delta)\,|_{\delta=0}\} = \frac{P_s+P_s^2}{nP_T}Var(r_s) + \frac{P_s+1}{n}Var(r_T) + \frac{P_s - P_s^2}{nP_T^2}R_s^2 - \frac{2P_s}{nP_T}R_sR_T.$

■

Appendix B

Details of the Mountain-Car Task

The mountain-car task (Figure 8) has two continuous state variables, the position of the car, p_t, and the velocity of the car, v_t. At the start of each trial, the initial state is chosen randomly, uniformly from the allowed ranges: $-1.2 \leq p \leq 0.5$, $-0.07 \leq v \leq 0.07$. The mountain geography is described by $altitude = \sin(3p)$. The action, a_t, takes on values in $\{+1, 0, -1\}$ corresponding to forward thrust, no thrust, and reverse thrust. The state evolution was according to the following simplified physics:

$$v_{t+1} = bound[v_t + 0.001a_t - g\cos(3p_t)]$$

and

$$p_{t+1} = bound\left[p_t + v_{t+1}\right],$$

where $g = -0.0025$ is the force of gravity and the *bound* operation clips each variable within its allowed range. If p_{t+1} is clipped in this way, then v_{t+1} is also reset to zero. Reward is −1 on all time steps. The trial terminates with the first position value that exceeds $p_{t+1} > 0.5$.

Notes

1. Arguably, yet a third mechanism for managing delayed reward is to change representations or world models (e.g., Dayan, 1993; Sutton, 1995).
2. In some previous work (e.g., Sutton & Barto, 1987, 1990) the traces were normalized by a factor of $1-\gamma\lambda$, which is equivalent to replacing the "1" in these equations by $1-\gamma\lambda$. In this paper, as in most previous work, we absorb this linear normalization into the step-size parameter, α, in equation (1).
3. The time index here is assumed to continue increasing across trials. For example, if one trial reaches a terminal state at time τ, then the next trial begins at time $\tau+1$.
4. For this reason, this estimate is sometimes also referred to as the *certainty equivalent* estimate (e.g., Kumar and Varaiya, 1986).
5. In theory it is possible to get this down to $O(n^{2.376})$ operations (Baase, 1988), but, even if practical, this is still far too complex for many applications.
6. Although this algorithm is indeed identical to TD(λ), the theoretical results for TD(λ) on stationary prediction problems (e.g., Sutton, 1988; Dayan, 1992) do not apply here because the policy is continually changing, creating a *nonstationary* prediction problem.
7. This is a very simple way of assuring initial exploration of the state space. Because most values are better than they should be, the learning system is initially disappointed no matter what it does, which causes it to try a variety of things even though its policy at any one time is deterministic. This approach was sufficient for this task, but of course we do not advocate it in general as a solution to the problem of assuring exploration.

References

Albus, J. S., (1981). *Brain, Behavior, and Robotics*, chapter 6, pages 139–179. Byte Books.

Baase, S., (1988). *Computer Algorithms: Introduction to design and analysis*. Reading, MA: Addison-Wesley.

Barnard, E., (1993). Temporal-difference methods and Markov models. *IEEE Transactions on Systems, Man, and Cybernetics*, *23*(2), 357–365.

Barto, A. G. & Duff, M., (1994). Monte Carlo matrix inversion and reinforcement learning. In *Advances in Neural Information Processing Systems 6*, pages 687–694, San Mateo, CA. Morgan Kaufmann.

Barto, A. G., Sutton, R. S., & Anderson, C. W., (1983). Neuronlike elements that can solve difficult learning control problems. *IEEE Trans. on Systems, Man, and Cybernetics*, *13*, 835–846.

Bellman, R. E., (1957). *Dynamic Programming*. Princeton, NJ: Princeton University Press.

Curtiss, J. H., (1954). A theoretical comparison of the efficiencies of two classical methods and a Monte Carlo method for computing one component of the solution of a set of linear algebraic equations. In Meyer, H. A. (Ed.), *Symposium on Monte Carlo Methods*, pages 191–233, New York: Wiley.

Dayan, P., (1992). The convergence of TD(λ) for general λ. *Machine Learning*, *8*(3/4), 341–362.

Dayan, P., (1993). Improving generalization for temporal difference learning: The successor representation. *Neural Computation*, *5*(4), 613–624.

Dayan, P. & Sejnowski, T., (1994). TD(λ) converges with probability 1. *Machine Learning*, *14*, 295–301.

Holland, J. H., (1986). *Escaping brittleness: The possibilities of general-purpose learning algorithms applied to parallel rule-based systems*, Volume 2 of *Machine Learning: An Artificial Intelligence Approach*, chapter 20. Morgan Kaufmann.

Jaakkola, T., Jordan, M. I., & Singh, S. P., (1994). On the convergence of stochastic iterative dynamic programming algorithms. *Neural Computation*, *6*(6), 1185–1201.

Jaakkola, T., Singh, S. P., & Jordan, M. I., (1995). Reinforcement learning algorithm for partially observable Markov decision problems. In *Advances in Neural Information Processing Systems 7*. Morgan Kaufmann

Klopf, A. H., (1972). Brain function and adaptive systems A heterostatic theory. Technical Report AFCRL-72-0164, Air Force Cambridge Research Laboratories, Bedford, MA.

Kumar, P. R. & Varaiya, P. P., (1986). *Stochastic Systems: Estimation, Identification, and Adaptive Control*. Englewood Cliffs, N.J.: Prentice Hall.

Lin, L. J., (1992). Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, *8*(3/4), 293–321.

Miller, W. T., Glanz, F. H., & Kraft, L. G., (1990). CMAC. An associative neural network alternative to backpropagation. *Proc. of the IEEE*, *78*, 1561–1567.

Moore, A. W., (1991). Variable resolution dynamic programming: Efficiently learning action maps in multivariate real-valued state-spaces. In *Machine Learning: Proceedings of the Eighth International Workshop*, pages 333–337, San Mateo, CA. Morgan Kaufmann.

Peng, J., (1993). *Dynamic Programming-based Learning for Control*. PhD thesis, Northeastern University.

Peng, J. & Williams, R. J., (1994). Incremental multi-step Q-learning. In *Machine Learning: Proceedings of the Eleventh International Conference*, pages 226–232. Morgan Kaufmann.

Rubinstein, R., (1981). *Simulation and the Monte Carlo method*. New York: John Wiley Sons.

Rummery, G. A. & Niranjan, M., (1994). On-line Q-learning using connectionist systems. Technical Report CUED/F-INFENG/TR 166, Cambridge University Engineering Dept.

Sutton, R. S., (1984). *Temporal Credit Assignment in Reinforcement Learning*. PhD thesis, University of Massachusetts, Amherst, MA.

Sutton, R. S., (1988). Learning to predict by the methods of temporal differences. *Machine Learning*, *3*, 9–44.

Sutton, R. S., (1995). TD models: Modeling the world at a mixture of time scales. In *Machine Learning: Proceedings of the Twelth International Conference*, pages 531–539. Morgan Kaufmann.

Sutton, R. S. & Barto, A. G., (1987). A temporal-difference model of classical conditioning. In *Proceedings of the Ninth Annual Conference of the Cognitive Science Society*, pages 355–378, Hillsdale, NJ. Erlbaum.

Sutton, R. S. & Barto, A. G., (1990). Time-derivative models of Pavlovian conditioning. In Gabriel, M. & Moore, J. W. (Eds.), *Learning and Computational Neuroscience*, pages 497–537. Cambridge, MA: MIT Press.

Sutton, R. S. & Singh, S. P., (1994). On step-size and bias in temporal-difference learning. In *Eighth Yale Workshop on Adaptive and Learning Systems*, pages 91–96, New Haven, CT.

Sutton, R. S. & Whitehead, S. D., (1993). Online learning with random representations. In *Machine Learning: Proceedings of the Tenth Int. Conference*, pages 314–321. Morgan Kaufmann.

Tesauro, G. J., (1992). Practical issues in temporal difference learning. *Machine Learning*, *8*(3/4), 257–277.

Tsitsiklis, J., (1994). Asynchronous stochastic approximation and Q-learning. *Machine Learning*, *16*(3), 185–202.

Wasow, W. R., (1952). A note on the inversion of matrices by random walks. *Math. Tables Other Aids Comput.*, *6*, 78–81.

Watkins, C. J. C. H., (1989). *Learning from Delayed Rewards*. PhD thesis, Cambridge Univ., Cambridge, England.

Wilson, S. W., (to appear). Classifier fitness based on accuracy. *Evolutionary Computation*.

Received November 7, 1994
Accepted March 10, 1995
Final Manuscript October 4, 1995