# Simulation-Based Optimization of Markov Reward Processes

Peter Marbach and John N. Tsitsiklis
Laboratory for Information and Decision Systems
M.I.T.
Cambridge, MA 02139
email: marbach@mit.edu, jnt@mit.edu

**Abstract**

We propose a simulation-based algorithm for optimizing the average reward in a Markov Reward Process that depends on a set of parameters. As a special case, the method applies to Markov Decision Processes where optimization takes place within a parametrized set of policies. The algorithm involves the simulation of a single sample path, and can be implemented on-line. A convergence result (with probability 1) is provided.

## 1 Introduction

Markov Decision Processes and the associated dynamic programming (DP) methodology [1, 14] provide a general framework for posing and analyzing problems of sequential decision making under uncertainty. DP methods rely on a suitably defined value function that has to be computed for every state in the state space. However, many interesting problems involve very large state spaces, which prohibits the application of DP. In addition, DP assumes the availability of an exact model, in the form of transition probabilities. In many practical situations, such a model is not available and one must resort to simulation or experimentation with an actual system. For all of these reasons, dynamic programming in its pure form, may be inapplicable. The efforts to overcome the aforementioned difficulties involve two main ideas:

(1) The use of simulation to estimate quantities of interest, thus avoiding model-based computations.

(2) The use of parametric representations to overcome the curse of dimensionality.

Parametric representations, and the associated algorithms, can be broadly classified into three main categories.

(1) One can use a parametric representation of the value function. When the main ideas from DP are combined with such parametric representations, one obtains methods that go under the names of "reinforcement learning" or "neuro-dynamic programming"; see [16, 4] for textbook expositions, as well as the references therein.

(2) In an alternative approach, one considers a class of policies described in terms of a parameter vector $\theta$. Simulation is employed to estimate the gradient of the performance metric with respect to $\theta$, and the policy is improved by updating $\theta$ in a gradient direction. Methods of this type have been extensively explored in the IPA (infinitesimal perturbation analysis) literature [6, 10]. Many of these methods are focused on special problem types and do not readily extend to general Markov Decision Processes.

(3) A third approach, involves a so-called "actor-critic" architecture, that includes parametrizations of the policy (actor) and of the value function (critic) [3]. At present, little is known about the theoretical properties of such methods (see, however, [15]).

This paper concentrates on methods based on policy parametrization and (approximate) gradient improvement, in the spirit of item (b) above. It is actually not restricted to Markov Decision Processes, but it also applies to general Markov Reward Processes that depend on a parameter vector $\theta$. Our first step is to obtain a method for estimating the gradient of the performance metric. In this connection, we note the "likelihood ratio" method of [9], which has this flavor, but does not easily lend itself to on-line updating of the parameter vector. We find an alternative approach, based on a suitably defined "differential reward function," to be more convenient. It relies on a gradient formula that has been presented in different forms in [5, 7, 8, 11]. We exploit a variant of this formula and develop a method that updates the parameter vector $\theta$ at every renewal time, in an approximate gradient direction. Furthermore, we show how to construct a truly on-line method that updates the parameter vector at each time step. In this respect, our work is closer to the methods described in [6] (that reference assumes, however, the availability of an IPA estimator, with certain guaranteed properties that are absent in our context) and in [11] (which, however, does not contain convergence results). Due to space constraints, we state here our results without providing the corresponding proofs. For a more detailed version of the paper we refer to [13].

## 2 Markov Reward Processes Depending on a Parameter

In this section, we present our general framework, make a few assumptions, and state some basic results that will be needed later.

We consider a discrete-time, finite-state Markov chain $\{i_n\}$ with state space $S = \{1, \ldots, N\}$, whose transition probabilities depend on a parameter vector $\theta \in \Re^K$, and are denoted by

$$p_{ij}(\theta) = P(i_n = j \mid i_{n-1} = i,\ \theta).$$

Whenever the state is equal to $i$, we receive a one-stage reward, that also depends on $\theta$, and is denoted by $g_i(\theta)$.

For every $\theta \in \Re^K$, let $P(\theta)$ be the stochastic matrix with entries $p_{ij}(\theta)$. Let $\mathcal{P} = \{P(\theta) \mid \theta \in \Re^K\}$ be the set of all such matrices, and let $\overline{\mathcal{P}}$ be its closure. Note that every element of $\overline{\mathcal{P}}$ is also a stochastic matrix and, therefore, defines a Markov chain on the same state space. We make the following assumptions.

**Assumption 1** *There exists a state $i^* \in S$ such the Markov chain corresponding to every $P \in \overline{\mathcal{P}}$ has a single recurrent class which contains $i^*$.*

We will often refer to the times that the state $i^*$ is visited as *renewal times.*

**Assumption 2** *For every $i, j \in S$, $p_{ij}(\theta)$ and $g_i(\theta)$ are bounded, twice differentiable, and have bounded first and second derivatives.*

The performance metric that we use to compare different policies is the average reward criterion $\lambda(\theta)$, defined by

$$\lambda(\theta) = \lim_{t\to\infty} \frac{1}{t} E_\theta \left[\sum_{k=0}^{t} g_{i_k}(\theta)\right].$$

Here, $i_k$ is the state at time $k$, and the notation $E_\theta[\cdot]$ indicates that the expectation is taken with respect to the distribution of the Markov chain with transition probabilities $p_{ij}(\theta)$. Under Assumption 1, the average reward $\lambda(\theta)$ is well defined for every $\theta$, and does not depend on the initial state. Furthermore, the balance equations

$$\sum_{i=1}^{N} \pi_i(\theta) p_{ij}(\theta) = \pi_j(\theta), \qquad j = 1, \ldots, N-1, \tag{1}$$

$$\sum_{i=1}^{N} \pi_i(\theta) = 1, \tag{2}$$

have a unique solution $\pi(\theta) = (\pi_1(\theta), \ldots, \pi_N(\theta))$, with $\pi_i(\theta)$ being the steady state probability of state $i$ under that particular value of $\theta$, and the average reward is equal to

$$\lambda(\theta) = \sum_{i=1}^{N} \pi_i(\theta) g_i(\theta). \tag{3}$$

### 2.1 The Gradient of $\lambda(\theta)$

For any $\theta \in \Re^K$ and $i \in S$, we define the *differential reward* $v_i(\theta)$ of state $i$ by

$$v_i(\theta) = E_\theta \left[\sum_{k=0}^{T-1} \big(g_{i_k}(\theta) - \lambda(\theta)\big) \mid i_0 = i\right], \tag{4}$$

where $T = \min\{k > 0 \mid i_k = i^*\}$ is the first future time that state $i^*$ is visited. With this definition, it is well known that $v_{i^*}(\theta) = 0$.

The following proposition gives an expression for the gradient of the average reward $\lambda(\theta)$, with respect to $\theta$. A related expression (in a somewhat different context) was given in [11], and a proof can be found in [5]. The latter reference does not consider the case where $g_i(\theta)$ depends on $\theta$, but the extension is immediate.

**Proposition 1** *Let Assumptions 1 and 2 hold. Then,*

$$\nabla\lambda(\theta) = \sum_{i\in S} \pi_i(\theta) \left(\nabla g_i(\theta) + \sum_{j\in S} \nabla p_{ij}(\theta) v_j(\theta)\right).$$

Equation (3) for $\lambda(\theta)$ suggests that $\nabla\lambda(\theta)$ could involve terms of the form $\nabla\pi_i(\theta)$, but the expression given by Proposition 1 involves no such terms. This property is very helpful in producing simulation-based estimates of $\nabla\lambda(\theta)$.

### 2.2 An Idealized Gradient Algorithm

Given that our goal is to maximize the average reward $\lambda(\theta)$, it is natural to consider gradient-type methods. If the gradient of $\lambda(\theta)$ could be computed exactly, we would consider a gradient algorithm of the form [2]

$$\theta_{k+1} = \theta_k + \gamma_k \nabla\lambda(\theta_k).$$

Alternatively, if we could use simulation to produce an unbiased estimate $h_k$ of $\nabla\lambda(\theta_k)$, we could then employ the approximate gradient iteration

$$\theta_{k+1} = \theta_k + \gamma_k h_k.$$

The convergence of such a method can be established if we use a diminishing stepsize sequence and make suitable assumptions on the estimation errors. Unfortunately, it does not appear possible to produce unbiased estimates of $\nabla\lambda(\theta)$ in a manner that is consistent with on-line implementation based on a single sample path. This difficulty is bypassed by the method developed in the next section.

## 3 The Simulation–Based Method

In this section, we develop a simulation-based algorithm that updates $\theta$ using estimates of the gradient $\nabla\lambda(\theta)$ which are obtained by simulating a single sample path. We will show that this algorithm retains the convergence properties of the idealized gradient method.

For technical reasons, we make the following assumption on the transition probabilities $p_{ij}(\theta)$. In Section 5, this assumption is revisited and we argue that it need not be restrictive.

**Assumption 3** *There exists a positive scalar $\epsilon$, such that for every $i, j \in S$, we have*

$$\text{either} \quad p_{ij}(\theta) = 0, \quad \forall\, \theta, \qquad\qquad \text{or} \quad p_{ij}(\theta) \geq \epsilon, \quad \forall\, \theta.$$

### 3.1 Estimation of $\nabla\lambda(\theta)$

Throughout this subsection, we assume that the parameter vector $\theta$ is fixed to some value. Let $\{i_n\}$ be a sample path of the corresponding Markov chain, possibly obtained through simulation. Let $t_m$ be the time of the $m$th visit at the recurrent state $i^*$. We refer to the sequence $i_{t_m} i_{t_m+1}, \ldots, i_{t_{m+1}}$ as the $m$th *renewal cycle*, and we define its *length* $T_m$ by

$$T_m = t_{m+1} - t_m.$$

For a fixed $\theta$, the random variables $T_m$ are independent identically distributed, and have a (common) finite mean, denoted by $E_\theta[T]$.

Our first step is to rewrite the formula for $\nabla\lambda(\theta)$ in the form

$$\nabla\lambda(\theta) = \sum_{i\in S} \pi_i(\theta) \left( \nabla g_i(\theta) + \sum_{j\in S_i} p_{ij}(\theta) \left( \frac{\nabla p_{ij}(\theta)}{p_{ij}(\theta)} v_j(\theta) \right) \right),$$

where $S_i = \{j \mid p_{ij}(\theta) > 0\}$. Estimating the term $\pi_i(\theta)\nabla g_i(\theta)$ through simulation is straightforward, assuming that we are able to compute $\nabla g_i(\theta)$ for any given $i$ and $\theta$. The other term can be viewed as the expectation of $v_j(\theta)\nabla p_{ij}(\theta)/p_{ij}(\theta)$, with respect to the steady-state probability $\pi_i(\theta)p_{ij}(\theta)$ of transitions from $i$ to $j$. Furthermore, the definition (4) of $v_j(\theta)$, suggests that if $t_m < n \leq t_{m+1} - 1$, and $i_n = j$, we can use

$$\tilde{v}_{i_n}(\theta, \tilde{\lambda}) = \sum_{k=n}^{t_{m+1}-1} \left( g_{i_k}(\theta) - \tilde{\lambda} \right), \tag{5}$$

to estimate $v_j(\theta)$, where $\tilde{\lambda}$ is some estimate of $\lambda(\theta)$. Note that $v_{i^*}(\theta) = 0$ and does not need to be estimated. For this reason, we let

$$\tilde{v}_{i_n}(\theta, \tilde{\lambda}) = 0, \qquad \text{if } n = t_m.$$

By accumulating the above described estimates over a renewal cycle, we are finally led to an estimate of $\nabla\lambda(\theta)$ given by

$$F_m(\theta, \tilde{\lambda}) = \sum_{n=t_m}^{t_{m+1}-1} \left( \tilde{v}_{i_n}(\theta, \tilde{\lambda}) \frac{\nabla p_{i_{n-1}i_n}(\theta)}{p_{i_{n-1}i_n}(\theta)} + \nabla g_{i_n}(\theta) \right). \tag{6}$$

Note that the denominator $p_{i_{n-1}i_n}(\theta)$ is always positive, since only transitions that have positive probability will be observed. Also, the random variables $F_m(\theta, \tilde{\lambda})$ are independent and identically distributed for different values of $m$, because the transitions during distinct renewal cycles are independent.

We define $f(\theta, \tilde{\lambda})$ to be the expected value of $F_m(\theta, \tilde{\lambda})$, namely,

$$f(\theta, \tilde{\lambda}) = E_\theta[F_m(\theta, \tilde{\lambda})]. \tag{7}$$

The following proposition confirms that the expectation of $F_m(\theta, \tilde{\lambda})$ is aligned with $\nabla\lambda(\theta)$, to the extent that $\tilde{\lambda}$ is close to $\lambda(\theta)$.

**Proposition 2** *We have*

$$f(\theta, \tilde{\lambda}) = E_\theta[T]\nabla\lambda(\theta) + G(\theta)(\lambda(\theta) - \tilde{\lambda}),$$

*where*

$$G(\theta) = E_\theta \left[ \sum_{n=t_m+1}^{t_{m+1}-1} (t_{m+1} - n) \frac{\nabla p_{i_{n-1}i_n}(\theta)}{p_{i_{n-1}i_n}(\theta)} \right]. \tag{8}$$

### 3.2 An Algorithm that Updates at Visits to the Recurrent State

We now use the approximate gradient direction provided by Proposition 2, and propose a simulation-based algorithm that performs updates at visits to the recurrent state $i^*$. We use the variable $m$ to index the times when the recurrent state $i^*$ is visited, and the corresponding updates. The form of the algorithm is the following. At the time $t_m$ that state $i^*$ is visited for the $m$th time, we have available a current vector $\theta_m$ and an average reward estimate $\tilde{\lambda}_m$. We then simulate the process according to the transition probabilities $p_{ij}(\theta_m)$ until the next time $t_{m+1}$ that $i^*$ is visited and update according to

$$\theta_{m+1} = \theta_m + \gamma_m F_m(\theta_m, \tilde{\lambda}_m), \tag{9}$$

$$\tilde{\lambda}_{m+1} = \tilde{\lambda}_m + \gamma_m \sum_{n=t_m}^{t_{m+1}-1} (g_{i_n}(\theta_m) - \tilde{\lambda}_m), \tag{10}$$

where $\gamma_m$ is a positive stepsize sequence (cf. Assumption 4 below). To see the rationale behind Eq. (10) note that the expected update direction for $\tilde{\lambda}$ is

$$E_\theta \left[ \sum_{n=t_m}^{t_{m+1}-1} (g_{i_n}(\theta) - \tilde{\lambda}) \right] = E_\theta[T](\lambda(\theta) - \tilde{\lambda}), \tag{11}$$

which moves $\tilde{\lambda}$ closer to $\lambda(\theta)$.

**Assumption 4** *The stepsizes $\gamma_m$ are nonnegative and satisfy*

$$\sum_{m=1}^{\infty} \gamma_m = \infty, \qquad \sum_{m=1}^{\infty} \gamma_m^2 < \infty.$$

Assumption 4 is satisfied, for example, if we let $\gamma_m = 1/m$. We can now state our main result.

**Proposition 3** *Let Assumptions 1-4 hold, and let $\{\theta_m\}$ be the sequence of parameter vectors generated by the above described algorithm. Then, $\lambda(\theta_m)$ converges and*

$$\lim_{m\to\infty} \nabla\lambda(\theta_m) = 0,$$

*with probability 1.*

### 3.3 A Heuristic Argument

In this subsection, we approximate the algorithm by a suitable ODE (as in [12]), and establish the convergence properties of the ODE. While this argument does not constitute a proof, it illustrates the rationale behind our convergence result.

We replace the update directions by their expectations under the current value of $\theta$. The update equations for $\theta$ and $\tilde{\lambda}$ take the form

$$\begin{aligned}
\theta_{m+1} &= \theta_m + \gamma_m f(\theta_m, \tilde{\lambda}_m), \\
\tilde{\lambda}_{m+1} &= \tilde{\lambda}_m + \gamma_m E_{\theta_m}[T](\lambda(\theta_m) - \tilde{\lambda}_m),
\end{aligned}$$

where $f(\theta, \tilde{\lambda})$ is given by Proposition 2. With an asymptotically vanishing stepsize, and after rescaling time, this deterministic iteration behaves similar to the following system of differential equations:

$$\dot{\theta}_t = \nabla\lambda(\theta_t) + \frac{G(\theta_t)}{E_{\theta_t}[T]}(\lambda(\theta_t) - \tilde{\lambda}_t), \qquad (12)$$

$$\dot{\tilde{\lambda}}_t = \lambda(\theta_t) - \tilde{\lambda}_t. \qquad (13)$$

Note that $\tilde{\lambda}_t$ and $\lambda(\theta_t)$ are bounded functions since the one-stage reward $g_i(\theta)$ is finite-valued and, therefore, bounded. We will now argue that $\tilde{\lambda}_t$ converges.

We first consider the case where the initial conditions satisfy $\tilde{\lambda}_0 \leq \lambda(\theta_0)$. We then claim that

$$\tilde{\lambda}_t \leq \lambda(\theta_t), \qquad \forall\, t > 0. \qquad (14)$$

Indeed, suppose that at some time $t_0$ we have $\tilde{\lambda}_{t_0} = \lambda(\theta_{t_0})$. If $\nabla\lambda(\theta_{t_0}) = 0$, then we are at an equilibrium point of the differential equations, and we have $\tilde{\lambda}_t = \lambda(\theta_t)$ for all subsequent times. Otherwise, if $\nabla\lambda(\theta_{t_0}) \neq 0$, then $\dot{\theta}_{t_0} = \nabla\lambda(\theta_{t_0})$, and $\dot{\lambda}(\theta_{t_0}) > 0$. At the same time, we have $\dot{\tilde{\lambda}}_{t_0} = 0$, and this implies that $\tilde{\lambda}_t < \lambda(\theta_t)$ for $t$ slightly larger than $t_0$. The validity of the claim (14) follows. As long as $\tilde{\lambda}_t \leq \lambda(\theta_t)$, $\tilde{\lambda}_t$ is nondecreasing and since it is bounded, it must converge.

Suppose now that the initial conditions satisfy $\tilde{\lambda}_0 > \lambda(\theta_0)$. As long as this condition remains true, $\tilde{\lambda}_t$ is nonincreasing. There are two possibilities. If this condition remains true for all times, then $\tilde{\lambda}(t)$ converges. If not, then there exists a time $t_0$ such that $\tilde{\lambda}_{t_0} = \lambda(\theta_{t_0})$, which takes us back to the previously considered case.

Having concluded that $\tilde{\lambda}_t$ converges, we can use Eq. (13) to argue that $\lambda(\theta_t)$ must also converge to the same limit. Then, in the limit, $\theta_t$ evolves according to $\dot{\theta}_t = \nabla\lambda(\theta_t)$, from which it follows that $\nabla\lambda(\theta_t)$ must go to zero.

### 3.4 Implementation Issues

In this subsection, we indicate an economical way of computing the direction $F_m(\theta, \tilde{\lambda})$ of update of the vector $\theta$.

Taking into account that $\tilde{v}_{i_{t_m}}(\theta, \tilde{\lambda}) = 0$, Eq. (6) can be written as

$$\begin{aligned}
F_m \quad & (\theta, \tilde{\lambda}) \\
&= \sum_{n=t_m+1}^{t_{m+1}-1} \tilde{v}_{i_n}(\theta, \tilde{\lambda}) \frac{\nabla p_{i_{n-1}i_n}(\theta)}{p_{i_{n-1}i_n}(\theta)} + \sum_{n=t_m}^{t_{m+1}-1} \nabla g_{i_n}(\theta) \\
&= \nabla g_{i^*}(\theta) + \sum_{k=t_m+1}^{t_{m+1}-1} \Big(\nabla g_{i_k}(\theta) + (g_{i_k}(\theta) - \tilde{\lambda}) z_k\Big),
\end{aligned}$$

where

$$z_k = \sum_{n=t_m+1}^{k} \frac{\nabla p_{i_{n-1}i_n}(\theta)}{p_{i_{n-1}i_n}(\theta)}, \qquad k = t_m+1, \ldots, t_{m+1}-1,$$

is a vector (of the same dimension as $\theta$) that becomes available at time $k$. It can be updated recursively, with

$$z_{t_m} = 0, \qquad (15)$$

and

$$z_{k+1} = z_k + \frac{\nabla p_{i_k i_{k+1}}(\theta)}{p_{i_k i_{k+1}}(\theta)}, \qquad k = t_m, \ldots, t_{m+1} - 2. \qquad (16)$$

In order to implement the algorithm, on the basis of the above equations, we only need to maintain in memory $2K + 1$ scalars, namely $\tilde{\lambda}$, and the vectors $\theta$, $z$.

## 4 Algorithms that Update at Every Time Step

We now propose a fully on-line algorithm that updates the parameter vector $\theta$ at each time step. To do that, we use the result of the preceding subsection, which allows us to break down the total update into a sum of incremental updates carried out at each time step.

At a typical time $k$, the state is $i_k$, and the values of $\theta_k$, $z_k$, and $\tilde{\lambda}_k$ are available from the previous iteration.

We update $\theta$ and $\tilde{\lambda}$ according to

$$\begin{aligned}\theta_{k+1} &= \theta_k + \gamma_k\Big(\nabla g_{i_k}(\theta_k) + (g_{i_k}(\theta_k) - \tilde{\lambda}_k)z_k\Big),\\ \tilde{\lambda}_{k+1} &= \tilde{\lambda}_k + \gamma_k(g_{i_k}(\theta_k) - \tilde{\lambda}_k).\end{aligned}$$

We then simulate a transition to the next state $i_{k+1}$ according to the transition probabilities $p_{ij}(\theta_{k+1})$, and finally update $z$ by letting

$$z_{k+1} = \begin{cases} 0, & \text{if } i_{k+1} = i^*,\\ z_k + \dfrac{\nabla p_{i_k i_{k+1}}(\theta_k)}{p_{i_k i_{k+1}}(\theta_k)}, & \text{otherwise.}\end{cases}$$

In order to prove convergence of this version of the algorithm, we will use an additional condition on the stepsizes.

**Assumption 5** *The stepsizes $\gamma_k$ are nonincreasing. Furthermore, there exists a positive integer $p$ and a positive scalar $A$ such that*

$$\sum_{k=n}^{n+t}(\gamma_n - \gamma_k) \leq A t^p \gamma_n^2, \qquad \forall\ n, t > 0.$$

Assumption 5 is satisfied, for example, if we let $\gamma_k = 1/k$. We have the following convergence result.

**Proposition 4** *Let Assumptions 1-5 hold, and let $\{\theta_k\}$ be the sequence of parameter vectors generated by the above described algorithm. Then, $\lambda(\theta_k)$ converges and*

$$\lim_{k\to\infty} \nabla\lambda(\theta_k) = 0,$$

*with probability 1.*

## 5 Markov Decision Processes

In this section, we indicate how to apply our methodology to Markov Decision Processes.

We consider a Markov Decision Processes [1, 14] with finite state space $S = \{1, \ldots, N\}$ and finite action space $U = \{1, \ldots, L\}$. At any state $i$, the choice of a control action $u \in U$ determines the probability $p_{ij}(u)$ that the next state is $j$. The immediate reward at each time step is of the form $g_i(u)$, where $i$ and $u$ is the current state and action, respectively.

A (*randomized*) *policy* is defined as a mapping

$$\mu : S \mapsto [0,1]^L,$$

with components $\mu_u(i)$ such that

$$\sum_{u\in U}\mu_u(i) = 1, \qquad \forall\ i \in S.$$

Under a policy $\mu$, and whenever the state is equal to $i$, action $u$ is chosen with probability $\mu_u(i)$, independent of everything else. If for every state $i$ there exists a single $u$ for which $\mu_u(i)$ is positive (and, therefore, unity), we say that we have a *pure* policy.

For problems involving very large state spaces, it is impossible to even describe an arbitrary randomized policy $\mu$. This leads us to consider policies described in terms of a parameter vector $\theta = (\theta_1, \ldots, \theta_K)$, whose dimension $K$ is tractably small. We are interested in a method that performs small incremental updates of the parameter $\theta$. A method of this type can work only if the policy has a smooth dependence on $\theta$, and this is the main reason why we choose to work with randomized policies.

We allow $\theta$ to be an arbitrary element of $\Re^K$. With every $\theta \in \Re^K$, we associate a randomized policy $\mu(\theta)$, which at any given state $i$ chooses action $u$ with probability $\mu_u(i,\theta)$. Naturally, we require that every $\mu_u(i,\theta)$ be nonnegative and that $\sum_{u\in U}\mu_u(i,\theta) = 1$. Note that the resulting transition probabilities are given by

$$p_{ij}(\theta) = \sum_{u\in U}\mu_u(i,\theta)p_{ij}(u), \tag{17}$$

and the expected reward per stage is given by

$$g_i(\theta) = \sum_{u\in U}\mu_u(i,\theta)g_i(u).$$

The objective is to maximize the average reward under policy $\mu(\theta)$, which is denoted by $\lambda(\theta)$. This is a special case of the framework of Section 2. We now discuss the various assumptions introduced in earlier sections.

In order to satisfy Assumption 1, it suffices to assume that there exists a state $i^*$ which is recurrent under every pure policy, a property which is satisfied in many interesting problems. In order to satisfy Assumption 2, it suffices to assume that $\mu_u(i,\theta)$, for all $i \in S$ and all $u \in U$, is twice differentiable (in $\theta$) and has bounded first and second derivatives. Finally, Assumption 3 is implied by the following condition.

**Assumption 6** *There exists some $\epsilon > 0$ such that $\mu_u(i,\theta) \geq \epsilon$ for every $i$, $u$, and $\theta$.*

Since $\sum_{u\in U}\mu_u(i,\theta) = 1$, for every $\theta$, we have $\sum_{u\in U}\nabla\mu_u(i,\theta) = 0$, and

$$\nabla g_i(\theta) = \sum_{u\in U}\nabla\mu_u(i,\theta)(g_i(u) - \lambda(\theta)).$$

Furthermore,

$$\sum_{j\in S}\nabla p_{ij}(\theta)v_j(\theta) = \sum_{j\in S}\sum_{u\in U}\nabla\mu_u(i,\theta)p_{ij}(u)v_j(\theta).$$

Using these relations in the formula for $\nabla\lambda(\theta)$ provided by Proposition 1, and after some rearranging, we obtain

$$\nabla\lambda(\theta) = \sum_{i\in S}\sum_{u\in U} \pi_i(\theta)\mu_u(i,\theta)q_{i,u}(\theta)\frac{\nabla\mu_u(i,\theta)}{\mu_u(i,\theta)},$$

where

$$\begin{aligned} q_{i,u}(\theta) &= (g_i(u) - \lambda(\theta)) + \sum_{j\in S} p_{ij}(u)v_j(\theta) \\ &= E_\theta\left[\sum_{k=0}^{T-1}(g_{i_k}(u_k) - \lambda(\theta)) \mid i_0 = i, u_0 = u\right], \end{aligned}$$

and where $i_k$ and $u_k$ is the state and control at time $k$. Thus, $q_{i,u}(\theta)$ is the differential reward if control action $u$ is first applied in state $i$, and policy $\mu(\theta)$ is followed thereafter. It is the same as Watkins' $Q$-factor [17], suitably modified for the average reward case.

From here on, we can proceed as in Section 3 and obtain an algorithm that updates $\theta$ at the times $t_m$ that state $i^*$ is visited. The form of the algorithm is

$$\begin{aligned} \theta_{m+1} &= \theta_m + \gamma_m F_m(\theta_m, \tilde\lambda_m), \\ \tilde\lambda_{m+1} &= \tilde\lambda_m + \gamma_m \sum_{n=t_m}^{t_{m+1}-1}(g_{i_n}(u_n) - \tilde\lambda_m), \end{aligned}$$

where

$$F_m(\theta_m, \tilde\lambda_m) = \sum_{n=t_m}^{t_{m+1}-1} \tilde q_{i_n,u_n}\frac{\nabla\mu_{u_n}(i_n,\theta_m)}{\mu_{u_n}(i_n,\theta_m)},$$

and

$$\tilde q_{i_n,u_n} = \sum_{k=n}^{t_{m+1}-1}(g_{i_k}(u_k) - \tilde\lambda_m).$$

Similar to Section 4, an on-line version of the algorithm is also possible. The convergence results of Sections 3 and 4 remain valid, with only notation changes in the proof.

**Acknowledgments:** This research was supported by a contract with Siemens AG, Munich, Germany, and by contract DMI-9625489 with the National Science Foundation.

## References

[1] D. P. Bertsekas, Dynamic Programming and Optimal Control, Athena Scientific, Belmont, MA, 1995.

[2] D. P. Bertsekas, Nonlinear Programming, Athena Scientific, Belmont, MA, 1995.

[3] A. Barto, R. Sutton, and C. Anderson, "Neuron-Like Elements that Can Solve Difficult Learning Control Problems," *IEEE Trans. on Systems, Man and Cybernetics*, Vol. 13, 1983, pp. 835-846.

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

[5] X. R. Cao and H. F. Chen, "Perturbation Realization, Potentials, and Sensitivity Analysis of Markov Processes," *IEEE Transactions on Automatic Control*, Vol. 42, 1997, pp. 1382-1393.

[6] E. K. P. Chong and P. J. Ramadage, "Stochastic Optimization of Regenerative Systems Using Infinitesimal Perturbation Analysis," *IEEE Trans. on Automatic Control*, Vol. 39, 1994, pp. 1400-1410.

[7] X. R. Cao and Y. W. Wan, "Algorithms for Sensitivity Analysis of Markov Systems through Potentials and Perturbation Realization," *to appear in IEEE Trans. on Control Systems Technology*, 1998.

[8] M. C. Fu and J. Hu, "Smoothed Perturbation Analysis Derivative Estimation for Markov Chains," *Operations Research Letters*, Vol. 15, 1994, pp. 241-251.

[9] P. W. Glynn, "Stochastic Approximation for Monte Carlo Optimization," *Proceedings of the 1986 Winter Simulation Conference*, 1986. pp. 285-289.

[10] Y. C. Ho and X. R. Cao, *Perturbation Analysis of Discrete Event Systems*, Kluwer Academic Publisher, Boston, MA, 1991.

[11] T. Jaakkola, S. P. Singh, and M. I. Jordan, "Reinforcement Learning Algorithm for Partially Observable Markov Decision Problems," *Advances in Neural Information Processing Systems*, Vol. 7, Morgan Kaufman, San Francisco, CA, 1995, pp. 345-352.

[12] L. Ljung, "Analysis of Recursive Stochastic Algorithms," *IEEE Trans. on Automatic Control*, Vol. 22, 1977, pp. 551-575.

[13] P. Marbach, and J. N. Tsitsiklis, "Simulation-Based Optimization of Markov Reward Processes", *Lab. for Info. and Decision Systems Report LIDS-P-2411*, Massachusetts Institute of Technology, Cambridge, MA, 1998.

[14] M. L. Puterman, Markov Decision Processes: Discrete Stochastic Dynamic Programming, Wiley, New York, NY, 1994.

[15] K.V. Rao, "Learning Algorithms for Markov Decision Processes," *Master Thesis*, Dept. of Computer Science and Automation, Indian Institute of Science, Banglore, India, 1996.

[16] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction, MIT Press, Cambridge, MA, 1998.

[17] C. Watkins, "Learning from Delayed Rewards," *Ph.D. Thesis*, Cambridge University, UK, 1989.