An Average-Reward Reinforcement Learning Algorithm for Computing Bias-Optimal Policies

Sridhar Mahadevan
Department of Computer Science and Engineering
University of South Florida
Tampa, Florida 33620
mahadeva@csee.usf.edu

Abstract

Average-reward reinforcement learning (ARL) is an undiscounted optimality framework that is generally applicable to a broad range of control tasks. ARL computes *gain-optimal* control policies that maximize the expected payoff per step. However, gain-optimality has some intrinsic limitations as an optimality criterion, since for example, it cannot distinguish between different policies that all reach an absorbing goal state, but incur varying costs. A more selective criterion is *bias optimality*, which can filter gain-optimal policies to select those that reach absorbing goals with the minimum cost. While several ARL algorithms for computing gain-optimal policies have been proposed, none of these algorithms can guarantee bias optimality, since this requires solving at least *two nested* optimality equations. In this paper, we describe a novel model-based ARL algorithm for computing bias-optimal policies. We test the proposed algorithm using an admission control queuing system, and show that it is able to utilize the queue much more efficiently than a gain-optimal method by learning bias-optimal policies.

Motivation

Recently, there has been growing interest in an undiscounted optimality framework called average reward reinforcement learning (ARL) (Boutilier & Puterman 1995; Mahadevan 1994; 1996a; Schwartz 1993; Singh 1994; Tadepalli & Ok 1994). ARL is well-suited to many cyclical control tasks, such as a robot avoiding obstacles (Mahadevan 1996a), an automated guided vehicle (AGV) transporting parts (Tadepalli & Ok 1994), and for process-oriented planning tasks (Boutilier & Puterman 1995), since the average reward is a good metric to evaluate performance in these tasks. However, one problem with the average reward criterion is that it is not sufficiently selective, both in goal-based tasks and tasks with no absorbing goals. Figure 1 illustrates the limitation of the average reward criterion on a simple two-dimensional grid-world task. Here, the learner is continually rewarded by $+10$ for reaching and staying in the absorbing goal state **G**, and is rewarded -1 in all non-goal states. Clearly, all control policies that reach the goal will have the same average reward. Thus, the average reward criterion cannot be used to select policies that reach absorbing goals in the shortest time.

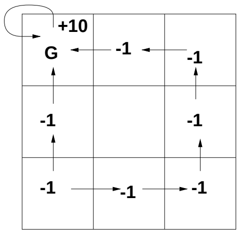

Figure 1: A simple grid-world navigation task to illustrate the unselectivity of the average-reward criterion. The two paths shown result in the same average reward ($+10$), but one takes three times as long to get to the goal **G**.

A more refined metric called *bias optimality* (Blackwell 1962) addresses the unselectivity of the average reward criterion. A policy is bias-optimal if it maximizes average reward (i.e. it is *gain-optimal*), and also maximizes the *average-adjusted* sum of rewards over all states. The latter quantity is simply the sum of rewards received, subtracting out the average reward at each step. For example, in Figure 1, the shorter path yields an average-adjusted reward of -22, whereas the longer path yields an average-adjusted reward of -66. Intuitively, bias optimality selects gain-optimal policies that maximize the average adjusted sum of rewards over the initial *transient* states (e.g., all non-goal states in Figure 1). In many practical problems where the average reward criterion is most useful, such as inventory control (Puterman 1994) and queueing systems (Kleinrock 1976), there may be several gain-optimal policies which can differ substantially in their "start-up" costs. In all such problems, it is critical to find bias-optimal policies.

While several ARL algorithms have been previously proposed (Schwartz 1993; Singh 1994; Tadepalli & Ok 1994), none of these algorithms will yield bias-optimal policies in general. In particular, while they can compute the bias-optimal policy for the simple grid-world task in Figure 1, they cannot discriminate the bias-optimal policy from the gain-optimal policy for the simple 3-state Markov decision process (MDP) given in Figure 2, or for the admission control queueing task shown in Figure 5. The main reason is these algorithms only solve one optimality equation, namely the average reward Bellman equation. It can be shown (Puterman 1994) that solving the Bellman equation alone is insufficient to determine bias-optimal policies, whenever there are several gain-optimal policies with different sets of recurrent states. The MDP's given in Figure 2 and Figure 5 fall into this category.

In this paper we propose a novel model-based ARL algorithm that is explicitly designed to compute bias-optimal policies. This algorithm is related to previous ARL algorithms but significantly extends them by solving *two nested* optimality equations to determine bias-optimal policies, instead of a single equation. We present experimental results using an admission control queuing system, showing that the new bias-optimal algorithm is able to learn to utilize the queue much more efficiently than a gain-optimal algorithm that only solves the Bellman equation.

Gain and Bias Optimality

We assune the standard Markov decision process (MDP) framework (Puterman 1994). An MDP consists of a (finite or infinite) set of states S, and a (finite or infinite) set of actions A for moving between states. In this paper we will assume that S and A are finite. We will denote the set of possible actions in a state x by $A(x)$. Associated with each action a is a state transition matrix $P(a)$, where $P_{xy}(a)$ represents the probability of moving from state x to y under action a. There is also a reward or payoff function $r : S \times A \rightarrow \mathcal{R}$, where $r(x,a)$ is the *expected* reward for doing action a in state x.

A *stationary deterministic policy* is a mapping $\pi : S \rightarrow A$ from states to actions. In this paper we consider only such policies, since a stationary deterministic bias-optimal policy exists. Two states x and y *communicate* under a policy π if there is a positive probability of reaching (through zero or more transitions) each state from the other. A state is *recurrent* under a policy π if starting from the state, the probability of eventually reentering it is 1. Note that this implies that recurrent states will be visited forever. A non-recurrent state is called *transient*, since at some finite point in time the state will never be visited again. A recurrent class of states is a set of recurrent states that all communicate with each other, and do not communicate with any state outside this class. An MDP is termed *unichain* if the transition matrix corresponding to every policy contains a single recurrent class, and a (possibly empty) set of transient states. Many interesting problems involve unichain MDP's, such as stochastic grid-world problems (Mahadevan 1996a), the admission control queueing system shown in Figure 5, and an AGV transporting parts (Tadepalli & Ok 1994).

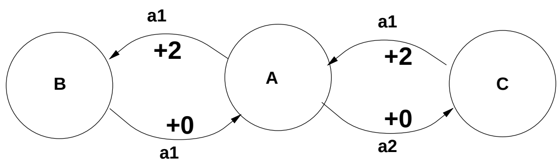

Figure 2: A simple 3-state MDP that illustrates the unselectivity of the average reward criterion in MDP's with no absorbing goal states. Both policies in this MDP are gain-optimal, however only the policy that selects action **a1** in state **A** is bias-optimal.

Average reward MDP aims to compute policies that yield the highest expected payoff per step. The average reward $\rho^\pi(x)$ associated with a particular policy π at a state x is defined as

$$\rho^\pi(x) = \lim_{N\rightarrow\infty} \frac{E\left(\sum_{t=0}^{N-1} R_t^\pi(x)\right)}{N}, \quad \forall x \in S,$$

where $R_t^\pi(x)$ is the reward received at time t starting from state x, and actions are chosen using policy π. $E(.)$ denotes the expected value. A *gain-optimal* policy π^* is one that maximizes the average reward over all states, that is, $\rho^{\pi^*}(x) \geq \rho^\pi(x)$ over all policies π and states x. Note that in unichain MDP's, the average reward of any policy is state independent. That is, $\rho^\pi(x) = \rho^\pi(y) = \rho^\pi, \quad \forall x, y \in S, \forall \pi$.

As shown in Figure 1, gain-optimality is not sufficiently selective in goal-based tasks, as well as in tasks with no absorbing goals. A more selective criterion called *bias optimality* addresses this problem. The *average adjusted* sum of rewards earned following a policy π (assuming an aperiodic MDP) is

$$V^\pi(s) = \lim_{N\rightarrow\infty} E\left(\sum_{t=0}^{N-1} (R_t^\pi(s) - \rho^\pi)\right),$$

where ρ^π is the average reward associated with policy π. A policy π^* is termed *bias-optimal* if it is gain-optimal, and it also maximizes the average-adjusted values, that is $V^{\pi^*}(x) \geq V^{\pi}(x)$ over all $x \in S$ and policies π. The relation between gain-optimal and bias-optimal policies is depicted in Figure 3.

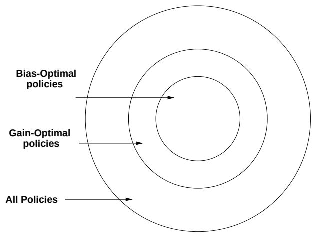

Figure 3: This diagram illustrates the relation between gain-optimal and bias-optimal policies.

In the example 3-state MDP in Figure 2, both policies are gain-optimal since they yield an average reward of 1. However, the policy π that selects action **a1** in state **A** generates bias values $V^{\pi}(A) = 0.5$, $V^{\pi}(B) = -0.5$, and $V^{\pi}(C) = 1.5$. The policy π is bias-optimal because the only other policy is π' that selects action **a2** in state **A**, and generates bias values $V^{\pi'}(A) = -0.5$, $V^{\pi'}(B) = -1.5$, and $V^{\pi'}(C) = 0.5$.

Bias-Optimality Equations

The key difference between gain and bias optimality is that the latter requires solving two nested optimality equations for a unichain MDP. The first equation is the well-known average-reward analog of Bellman's optimality equation.

Theorem 1 *For any MDP that is either unichain or communicating, there exists a value function V^* and a scalar ρ^* satisfying the equation over all states*

$$V^*(x) + \rho^* = \max_{a \in A(x)} \left(r(x,a) + \sum_y P_{xy}(a) V^*(y) \right), \tag{1}$$

such that the greedy policy π^* resulting from V^* achieves the optimal average reward $\rho^* = \rho^{\pi^*}$ where $\rho^{\pi^*} \geq \rho^{\pi}$ over all policies π.

Here, "greedy" policy means selecting actions that maximize the right hand side of the above Bellman equation. There are many algorithms for solving this equation, ranging from DP methods (Puterman 1994) to ARL methods (Schwartz 1993). However, solving this equation does not suffice to discriminate between bias-optimal and gain-optimal policies for a unichain MDP. In particular, none of the previous ARL algorithms can discriminate between the bias-optimal policy and the gain-optimal policy for the 3-state MDP in Figure 2. A second optimality equation has to be solved to determine the bias-optimal policy.

Theorem 2 *Let V be a value function and ρ be a scalar that together satisfy Equation 1. Define $A_V(i) \subseteq A(i)$ to be the set of actions that maximize the right-hand side of Equation 1. There exists a function $W : S \rightarrow \mathcal{R}$ satisfying the equation over all states*

$$W(x) = \max_{a \in A_V(x)} \left(-V(x) + \sum_y P_{xy}(a) W(y) \right), \tag{2}$$

such that any policy formed by choosing actions in A_V that maximize the right-hand side of the above equation is bias-optimal.

These optimality equations are nested, since the set A_V of actions over which the maximization is sought in Equation 2 is restricted to those that maximize the right-hand side of Equation 1. The function W, which we will refer to as the *bias offset*, holds the key to policy improvement, since it indicates how close a policy is to achieving bias-optimality.

A Model-based Bias-Optimality Algorithm

We now describe a model-based algorithm for computing bias-optimal policies for a unichain MDP. The algorithm estimates the transition probabilities from online experience, similar to (Jalali & Ferguson 1989; Tadepalli & Ok 1994). However, unlike these previous algorithms, the proposed algorithm solves both optimality equations (Equation 1 and Equation 2 above). Since the two equations are nested, one possibility is to solve the first equation by successive approximation, and then solve the second equation. However, stopping the successive approximation process for solving the first equation at any point will result in some finite error, which could prevent the second equation from being solved. A better approach is to interleave the successive approximation process and solve both equations simultaneously (Federgruen & Schweitzer 1984).

The bias optimality algorithm is described in Figure 4. The transition probabilities $P_{ij}(a)$ and expected rewards $r(i,a)$ are inferred online from actual transitions (steps 7 through 10). The set $h(i)$ represents all actions in state i that maximize the right-hand side of Equation 1 (step 3). The set $w(i)$, on the other hand, refers to the subset of actions in $h(i)$ that also maximize the right-hand side of Equation 2. The algorithm successively computes $A(i, \epsilon_n)$ and $w(i, \epsilon_n)$, the set of gain-optimal actions that are within ϵ_n of the maximum value, and the set of bias-optimal actions within this gain-optimal set. This allows the two nested equations to be solved simultaneously. Here, ϵ_n is any series of real numbers that slowly decays as $n \rightarrow \infty$, similar to a "learning rate". Note that since the algorithm is based on stochastic approximation, some residual error is unavoidable, and thus ϵ_n should be decayed only up to some small value > 0.

The algorithm normalizes the bias values and bias offset values by grounding these quantities to 0 at a reference state. This normalization bounds these two quantities, and also improves the numerical stability of average reward algorithms (Puterman 1994). In the description, we have proposed choosing the reference state that is recurrent under all policies, if such exists, and is known beforehand. For example, in a standard stochastic grid-world problem (Mahadevan 1996a), the goal state satisfies this condition. In the admission control queueing task in Figure 5, the state $(0, 0)$ satisfies this condition. The policy output by the algorithm maximizes the expected bias offset value, which, as we discussed above, is instrumental in policy improvement.

Bias Optimality in an Admission Control Queuing System

We now present some experimental results of the proposed bias optimality algorithm using an admission control system, which is a well-studied problem in queueing systems (Kleinrock 1976). Generally speaking, there are a number of *servers*, each of which provides service to *jobs* that are arriving continuously according to some distribution. In this paper, for the sake of simplicity, we assume the M/M/1 queuing model, where the arrivals and service times are independent, memoryless, and distributed exponentially, and there is only 1 server. The arrival rate is modeled by parameter λ, and the service rate by parameter μ. At each arrival, the queue controller has to decide whether to *admit* the new job into the queue, or to *reject* it. If admitted, each job immediately generates a fixed reward R for the controller, which also incurs a *holding cost* $f(j)$ for the j jobs currently being serviced.

The aim is to infer an optimal policy that will maximize the rewards generated by admitting new jobs, and simultaneously minimize the holding costs of the existing jobs in the queue. Stidham(Stidham 1978) proved that if the holding cost function $f(j)$ is convex and nondecreasing, a *control limit* policy is optimal. A control limit policy is one where an arriving new job is admitted into the queue if and only if there are fewer than L jobs in the system. Recently, Haviv and Puterman (Haviv & Puterman) show that if the cost function $f(j) = cj$, there are at most *two* gain-optimal control limit policies, namely admit L and admit $L + 1$, but only one of them is also bias-optimal (admit $L+1$). Intuitively, the admit $L+1$ policy is bias-optimal because the additional cost of the new job is offset by the extra

1. **Initialization:** Let $n = 0$, bias function $V(x) = 0$, and bias-offset function $W(x) = 0$. Let the initial state be i. Initialize $N(i, a) = 0$, the number of times action a has been tried in state i. Let $T(i, a, k) = 0$, the number of times action a has caused a transition from i to k. Let the expected rewards $r(i, a) = 0$. Let s be some reference state, which is recurrent under all policies.
2. Let $H(i, a) = r(i, a) + \sum_j P_{ij}(a)V(j), \ \forall a \in A(i)$.
3. Let $h(i) = \{a \in A(i) | a \text{ maximizes } H(i, a)\}$. Let $A(i, \epsilon_n)$ be the set of actions that are within ϵ_n of the maximum $H(i, a)$ value.
4. Let $w(i, \epsilon_n) = \{a \in A(i, \epsilon_n) | a$ maximizes $\sum_j P_{ij}(a)W(j)\}$.
5. With probability $1-p_{exp}$, select action A to be some $a_o \in w(i, \epsilon_n)$. Otherwise let action A be any random action $a_r \in A(i)$.
6. Carry out action A. Let the next state be k, and immediate reward be $\hat{r}(i, a)$.
7. $N(i, A) \leftarrow N(i, A) + 1$.
8. $T(i, A, k) \leftarrow T(i, A, k) + 1$.
9. $P_{ik}(A) \leftarrow \frac{T(i,A,k)}{N(i,A)}$.
10. $r(i, A) \leftarrow r(i, A)(1 - \frac{1}{N(i,A)}) + \frac{1}{N(i,A)}\hat{r}(i, a)$.
11. $V(i) \leftarrow \max_{a \in A(i)}(H(i, a)) - \max_{a \in A(s)}(H(s, a))$.
12. $W(i) \leftarrow \max_{a \in A(i,\epsilon_n)} \left(\sum_j P_{ij}(a)W(j) - V(i)\right) - \max_{a \in A(s,\epsilon_n)} \left(\sum_j P_{sj}(a)W(j) - V(s)\right)$.
13. If $n < MAX\_STEPS$, set $n \leftarrow n + 1$, and $i \leftarrow k$ and go to step 5.
14. Output $\pi(i) \in w(i, \epsilon_n)$.

Figure 4: A model-based algorithm for computing bias-optimal policies for unichain MDP's.

reward received. Note that since rejected jobs never return, a policy that results in a larger queue length is better than one that results in a smaller queue length, provided the average reward of both policies are equal.

Since the M/M/1 queuing model is a continuous time MDP, we first convert it by *uniformization* (Puterman 1994) into a discrete time MDP. Figure 5 illustrates the general structure of the uniformized M/M/1 admission control queuing system. States in the figure are pairs (s, j), where s represents the number of jobs currently in the queue, and j is a boolean-valued variable indicating whether a new job has arrived. In states $(s, 1)$, there are two possible actions, namely reject the new job ($a = 0$), or admit the new job ($a = 1$). In states $(s, 0)$, there is only one possible action, namely continue the process ($a = 0$). Theoretically, there is an infinite number of states in the system, but in practice, a finite upper bound needs to be imposed on the queue size. Note that the two gain-optimal policies (admit L and admit $L+1$) have different sets of recurrent states, just as the 3-state MDP in Figure 2. States $(0, x)$ to $(L-1, x)$ form a recurrent class in the admit L policy, whereas states $(0, x)$ to (L, x) form the recurrent class in the admit $L+1$ policy.

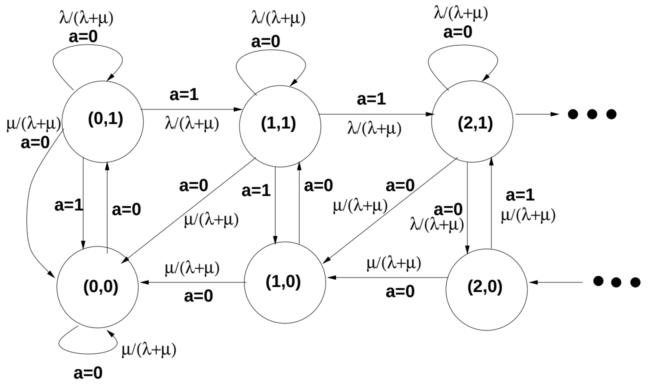

Figure 5: This diagram illustrates the MDP representation of the uniformized M/M/1 admission control queuing system for the average reward case.

The reward function for the average reward version of the admission control queuing system is as follows. If there are no jobs in the queue, and no new jobs have arrived, the reward is 0. If a new job has arrived and admitted in state s, the reward equals to the difference between the fixed payoff R for admitting the job and the cost of servicing the $s+1$ resulting jobs in the queue. Finally, if the job is not admitted, the reward is the service cost of the existing s jobs. There is an additional multiplicative term $\lambda + \mu$ that results from the uniformization process.

$$\begin{aligned} r((0,0),0) &= r((0,1),0) = 0. \\ r((s,1),1) &= [R - f(s+1)](\lambda+\mu), \quad s \geq 0. \\ r((s,0),0) &= r((s,1),0) = -f(s)(\lambda+\mu), \quad s \geq 1. \end{aligned}$$

Table 1 compares the performance of the bias-optimal algorithm with a simplified gain-optimal algorithm for several sets of parameters for the admission control system. We selected these from a total run of around 600 parameter combinations since these produced the largest improvements. Each combination was tested for 30 runs, with each run lasting 200,000 steps. Of these 600 parameter sets, we observed improvements of 25% or more in a little over 100 cases. In all other cases, the two algorithms performed equivalently, since they yielded the same average reward and average queue length. In every case shown in the table, there is substantial improvement in the performance of the bias-optimal algorithm, as measured by the increase in the average size of the queue. What this means in practice is that the bias-optimal algorithm allows much better utilization of the queue, without increasing the cost of servicing the additional items in the queue. Note that the improvement will occur whenever there are multiple gain-optimal policies, only one of which is bias-optimal. If there is only one gain-optimal policy, the bias optimality algorithm will choose that policy and thus perform as well as the gain-optimal algorithm.

λ	μ	R	c	Queue Size Increase
5	5	12	1	96.4
3	4	15	3	73.2
4	5	15	3	71.8
3	4	21	4	68.9
5	5	15	1	61.3
5	4	21	1	51.2
5	5	24	1	49.1
1	1	12	1	48.4
4	4	12	1	48.0
2	2	15	1	47.9

Table 1: This table compares the performance of the model-based bias-optimal algorithm with a (gain-optimal) simplification of the same algorithm that only solves the Bellman equation.

Related Work

To our knowledge, the proposed bias optimality algorithm represents the first ARL method designed explicitly for bias optimality. However, several previous algorithms exist in the DP and OR literature. These range from policy iteration (Veinott 1969; Puterman 1994) to linear programming (Denardo 1970). Finally, Federgruen and Schweitzer (Federgruen & Schweitzer 1984) study successive approximation methods for solving a general sequence of nested optimality equations, such as Equation 1 and Equation 2. We expect that bias-optimal ARL algorithms, such as the one described in this paper, will scale better than these previous non-adaptive bias-optimal algorithms. Bias-optimal ARL algorithms also have the added benefit of not requiring detailed knowledge of the particular MDP. However, these previous DP and OR algorithms are provably convergent, whereas we do not yet have a convergence proof for our algorithm.

Future Work

This paper represents the first step in studying bias optimality in ARL. Among the many interesting issues to be explored are the following:

- *Model-free Bias Optimality Algorithm:* We have also developed a model-free bias optimality algorithm (Mahadevan 1996b), which extends previous model-free ARL algorithms, such as R-learning (Schwartz 1993), to compute bias optimal policies by solving both optimality equations.

- *Scale-up Test on More Realistic Problems:* In this paper we only report experimental results on an admission control queuing domain. We propose to test our algorithm on a wide range of other problems, including more generalized queuing systems (Kleinrock 1976) and robotics related tasks (Mahadevan 1996a).

Acknowledgements

I am indebted to Martin Puterman for many discussions regarding bias optimality. I thank Larry Hall, Michael Littman, and Prasad Tadepalli for their detailed comments on this paper. I also thank Ken Christensen for helping me understand queueing systems. This research is supported in part by an NSF CAREER Award Grant No. IRI-9501852.

References

Blackwell, D. 1962. Discrete dynamic programming. *Annals of Mathematical Statistics* 33:719–726.

Boutilier, C., and Puterman, M. 1995. Process-oriented planning and average-reward optimality. In *Proceedings of the Fourteenth JCAI*, 1096–1103. Morgan Kaufmann.

Denardo, E. 1970. Computing a bias-optimal policy in a discrete-time Markov decision problem. *Operations Research* 18:272–289.

Federgruen, A., and Schweitzer, P. 1984. Successive approximation methods for solving nested functional equations in Markov decision problems. *Mathematics of Operations Research* 9:319–344.

Haviv, M., and Puterman, M. Bias optimality in controlled queueing systems. To Appear in Journal of Applied Probability.

Jalali, A., and Ferguson, M. 1989. Computationally efficient adaptive control algorithms for Markov chains. In *Proceedings of the 28th IEEE Conference on Decision and Control*, 1283–1288.

Kleinrock, L. 1976. *Queueing Systems*. John Wiley.

Mahadevan, S. 1994. To discount or not to discount in reinforcement learning: A case study comparing R-learning and Q-learning. In *Proceedings of the Eleventh International Conference on Machine Learning*, 164–172. Morgan Kaufmann.

Mahadevan, S. 1996a. Average reward reinforcement learning: Foundations, algorithms, and empirical results. *Machine Learning* 22:159–196.

Mahadevan, S. 1996b. Sensitive-discount optimality: Unifying average-reward and discounted reinforcement learning. In *Proceedings of the 13th International Conference on Machine Learning*. Morgan Kaufmann. To Appear.

Puterman, M. 1994. *Markov Decision Processes: Discrete Dynamic Stochastic Programming*. John Wiley.

Schwartz, A. 1993. A reinforcement learning method for maximizing undiscounted rewards. In *Proceedings of the Tenth International Conference on Machine Learning*, 298–305. Morgan Kaufmann.

Singh, S. 1994. Reinforcement learning algorithms for average-payoff Markovian decision processes. In *Proceedings of the 12th AAAI*. MIT Press.

Stidham, S. 1978. Socially and individually optimal control of arrivals to a GI/M/1 queue. *Management Science* 24(15).

Tadepalli, P., and Ok, D. 1994. H learning: A reinforcement learning method to optimize undiscounted average reward. Technical Report 94-30-01, Oregon State Univ.

Veinott, A. 1969. Discrete dynamic programming with sensitive discount optimality criteria. *Annals of Mathematical Statistics* 40(5):1635–1660.