Sensitive Discount Optimality: Unifying Discounted and Average Reward Reinforcement Learning

Sridhar Mahadevan *
Department of Computer Science and Engineering
University of South Florida
Tampa, Florida 33620
Tel: (813) 974-3260
mahadeva@csee.usf.edu

March 12, 1996

Abstract

Research in reinforcement learning (RL) has thus far concentrated on two optimality criteria: the discounted framework, which has been very well-studied, and the average-reward framework, in which interest is rapidly increasing. In this paper, we present a framework called *sensitive discount optimality* which offers an elegant way of linking these two paradigms. Although sensitive discount optimality has been well studied in dynamic programming, with several provably convergent algorithms, it has not received any attention in RL. This framework is based on studying the properties of the expected cumulative discounted reward, as discounting tends to 1. Under these conditions, the cumulative discounted reward can be expanded using a Laurent series expansion to yields a sequence of terms, the first of which is the average reward, the second involves the average adjusted sum of rewards (or bias), etc. We use the sensitive discount optimality framework to derive a new model-free average reward technique, which is related to Q-learning type methods proposed by Bertsekas, Schwartz, and Singh, but which unlike these previous methods, optimizes both the first and second terms in the Laurent series (average reward and bias values).

Keywords: Reinforcement learning, Average Reward.
Statement: This paper has not been submitted to any other conference.

*This research is supported by a National Science Foundation CAREER Award Grant No. IRI-9501852. This paper is to appear in the 13th International Conference on Machine Learning (IMLC-96), Bari, Italy, July 3rd-6th.

1 Motivation

Reinforcement learning (RL) has become one of the most actively studied areas in machine learning [7, 20]. Thus far, the work in RL has concentrated on one of two optimality criteria. The first is the traditional *discounted* framework, which has been extremely well-studied with several provably convergent algorithms such as Q-learning [24] and TD(λ) [18]. However, recently, there has also been growing interest in a second undiscounted optimality framework called average reward reinforcement learning (ARL) [1, 3, 11, 12, 15, 16, 21].

In this paper, we present an elegant framework called *sensitive discount optimality* [2, 14, 23] that links the discounted and average reward paradigms. The sensitive discount optimality framework is hardly new, with its roots dating back to Blackwell's classic paper [2], and there is a fairly extensive literature on this topic in the dynamic programming (DP) and operations research (OR) areas (Puterman's recent book [14] provides a detailed bibliography). Sensitive discount optimality is based on studying the properties of the expected cumulative discounted reward, as discounting tends to 1. Under these conditions, Blackwell [2] showed that the cumulative discounted reward can be expanded using a Laurent series expansion to yield a sequence of terms, the first of which involves the average reward, the second involves the average adjusted sum of rewards (or *bias* value), etc. However, to date, the sensitive discount optimality framework has not received much attention in the RL community (however, see [12]).

In this paper we use the sensitive discount optimality framework to derive a new model-free average-reward RL method that is demonstrably superior to previous methods, such as R-learning [15] and its variants [1, 16]. In particular, the new model-free algorithm is not only able to find policies that maximize the average reward (i.e. *gain-optimal*), but is also able to prune these to find more selective *bias-optimal* policies that maximize the average-adjusted sum of rewards over all states. Bias optimality is crucial in tasks with absorbing goal states, such as the well-known grid world navigation task [19], as well as in tasks that do not have any absorbing goals, such as queueing [5, 8, 17].

Schwartz [15] used the term *T-optimality* to refer to bias optimality, and erroneously assumed that R-learning could find bias-optimal policies. However, finding bias-optimal policies requires solving in general 3 nested optimality equations, and at least 2 nested equations for a unichain MDP (a special type of problem where the average reward is the same over all states). Since R-learning (and all other previous average reward RL methods [16, 21]) only solve the average reward Bellman equation, they cannot be guaranteed to find bias-optimal policies even in the restricted case of unichain MDP's.

The rest of this paper is organized as follows. Section 2 briefly reviews the existing discounted and average reward optimality frameworks. Section 3 presents the sensitive discount optimality framework that generalizes the average and discounted frameworks. Section 4 presents a series of nested optimality equations that is needed to compute sensitive discount optimal policies. In Section 5, we focus on bias optimality, and present two nested optimality equations for this special case. Section 6 presents a model-free RL algorithm for bias-optimal policies that extends previous average reward RL methods. Section 7 describes some experiments comparing the two methods on a simple queueing task. Finally, Section 8

discusses some issues for future research.

2 Discounted and Average Reward Optimality

We begin by reviewing the two existing optimality criteria: discounted and average reward optimality. We start with the familiar Markov Decision Process (MDP) framework underlying RL [1, 14]. 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 $r : S \times A \to \mathcal{R}$, where $r(x, a)$ is the *expected* reward for doing action a in state x.

A *stationary deterministic policy* is a mapping $\pi : S \to A$ from states to actions. In this paper we restrict our discussion to algorithms for such policies, since we are guaranteed that such policies exist for the sensitive discount optimality criterion. Any policy induces a state transition matrix P^π, where $P^\pi_{xy} = P_{xy}(\pi(x))$. Thus, any policy yields a Markov chain (S, P^π).

Discounted MDP: A *value* function $V : S \to \mathcal{R}$ is any mapping from states to real numbers. In the discounted framework, a policy π is measured by a value function representing the cumulative discounted sum of rewards earned following that policy from any state s:

$$V^\pi_\gamma(s) = \lim_{N\to\infty} E\left(\sum_{t=0}^{N-1} \gamma^t R^\pi_t(s)\right),$$

where $\gamma < 1$ is the discount factor, and $R^\pi_t(s)$ is the reward received at time t starting from state s under policy π. $E(.)$ denotes expected value. An *optimal discounted policy* π^* maximizes the above value function over all states x and policies π, i.e. $V^{\pi^*}_\gamma(x) \geq V^\pi_\gamma(x)$. Q-learning [24] is a well-known RL algorithm that, under certain conditions, can be proved to converge to the optimal discounted policy [22]. However, there are a number of problems with discounting. As the discount factor $\gamma \to 1$, convergence of the algorithms typically slows down [12, 15]. Another problem is that the optimal policy often varies as the discount factor is changed. Finally, for many tasks, such as a robot avoiding obstacles [13], an AGV sorting parts on conveyer belts [21], or for process-oriented planning [3], the average reward seems a more natural optimality metric.

Average Reward MDP: The goal of average reward MDP is 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

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

where $R_t^\pi(x)$ is the reward received at time t starting from state x, and actions are chosen using policy π. 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.

A key limitation of gain optimality is that it cannot distinguish between policies which have the same average reward, but which incur very different "startup" costs over the initial transient states. Such situations occur in goal-based tasks, as well as in tasks with no absorbing goals. For example, in a standard grid world domain [19] where the goal is absorbing, all paths that lead to the goal yield the same average reward, but can incur very different initial costs over the non-goal transient states. A more selective criterion called *bias optimality* is needed to address this problem. To define bias optimality, we need to evaluate policies using a different value function, namely the *average adjusted* sum of rewards earned following a policy,

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

where ρ^π is the average reward associated with policy π. [1] The term $V^\pi(s)$ is usually referred to as the *bias* value.

A policy π^* is termed *bias-optimal* if it is gain-optimal, and it also maximizes bias values, that is $V^{\pi^*}(x) \geq V^\pi(x)$ over all $x \in S$ and policies π. The notion of bias-optimality was introduced by Blackwell [2], who also pioneered the study of average reward MDP as the limiting case of the discounted framework. It can be shown that a stationary bias-optimal policy is guaranteed to exist for any finite MDP [14]. We next show how gain and bias optimality are instances of a more general optimality criterion called sensitive discount optimality.

3 Sensitive Discount Optimality

Sensitive discount optimality [2, 14, 23] is based on studying the cumulative discounted sum of rewards, as the discount factor γ approaches 1. Since the cumulative discounted sum of rewards may be unbounded for $\gamma = 1$, the key idea here is to look for policies that remain optimal as $\gamma \to 1$. Blackwell [2] showed that when $\gamma \to 1$, the cumulative discounted sum of rewards received by following any policy π, i.e. V_γ^π, can be expanded using a Laurent series expansion to obtain a sequence of terms, the first of which represents the *average reward* ρ^π of the policy π, the second of which represents the *average adjusted* sum of rewards V^π, and so on.

Theorem 1 *(Blackwell) For any finite MDP, as $\gamma \to 1$, the value function V_γ^π induced by any policy π can be written as the following Laurent series*

$$V_\gamma^\pi(x) = \frac{1}{\gamma} \sum_{n=-1}^{\infty} (\frac{1-\gamma}{\gamma})^n y_n^\pi(x), \quad \forall x \in S, \tag{1}$$

[1] This limit assumes that all policies are aperiodic. For periodic policies, we need to use the Cesaro limit $V^\pi(s) = \lim_{N\to\infty} \frac{\sum_{k=0}^{N-1} E\left(\sum_{t=0}^{k} (R_t^\pi(s) - \rho^\pi)\right)}{N}$.

where the first two terms in the series are $y^{\pi}_{-1}(x) = \rho^{\pi}(x)$, *and* $y^{\pi}_{0}(x) = V^{\pi}(x)$, *and the subsequent terms in the series are* $y^{\pi}_{n}(x) = (-1)^n H^{n+1}_{P^{\pi}} r(x, \pi(x))$, *where* $H_P = (I - P + P^*)^{-1}(I - P^*)$ *is called the* deviation matrix, *and* P^* *is the limiting matrix for* P.

Blackwell also introduced a more refined notion of optimality called *Blackwell optimality*. A policy π^* is Blackwell optimal if there exists a $0 \leq \gamma^* < 1$ such that

$$V^{\pi^*}_{\gamma}(s) \geq V^{\pi}_{\gamma}(s) \quad \forall \gamma \geq \gamma^*, \gamma < 1, \quad \forall \pi, \quad \forall x \in S$$

Intuitively, a Blackwell optimal policy π^* is one that remains optimal for all values of the discount factor $\geq \gamma^*$, but less than 1. Blackwell proved that there always exists a stationary deterministic policy that satisfies this metric for finite state and action spaces. As we will see in the next section, the Laurent series can be used to develop a *nested* set of optimality equations for computing a Blackwell optimal policy.

To see how Blackwell optimality unifies the discounted and average reward frameworks, we need to define another sensitive discount optimality metric called *n-discount-optimality*, which is due to Veinott [23]:

Definition 1 *A policy* π^* *is* n-discount-optimal, *for* $n = -1, 0, 1, ...$, *if for each state* s, *and over all policies* π,

$$\lim_{\gamma \to 1} (1 - \gamma)^{-n} \left(V^{\pi^*}_{\gamma}(s) - V^{\pi}_{\gamma}(s) \right) \geq 0.$$

We now consider some special cases of this general optimality metric to illustrate how it nicely generalizes the optimality metrics used previously in both standard discounted RL and average reward RL. Detailed proofs of the following equivalences can be found in [14].

Theorem 2 *(Veinott)* π^* *is a* -1-discount-optimal *policy if and only if* π^* *is a gain-optimal policy.*

Theorem 3 *(Veinott)* π^* *is a* 0-discount-optimal *policy if and only if* π^* *is a bias-optimal (or T-optimal) policy.*

These two theorems provide the link between n-discount-optimality and the average reward framework. Figure 1 shows a simple MDP that illustrates the concept of n-discount-optimality, and relates it to Blackwell optimality.

Figure 2 illustrates the structure of *n-discount-optimal* policies. The most selective optimality criterion is Blackwell optimality, and corresponds to a policy π^* that is *n-discount-optimal* for all $n \geq -1$. It is interesting to note that a policy that is *m-discount-optimal* will also be *n-discount-optimal* for all $n < m$. Thus, bias optimality is a more selective optimality metric than gain optimality. This suggests a computational procedure for producing Blackwell-optimal policies, by starting with *-1-discount-optimal* policies, and successively pruning these to produce *i-discount-optimal* policies, for $i > -1$. Exactly such a policy iteration procedure for multichain MDP's is described in [14].

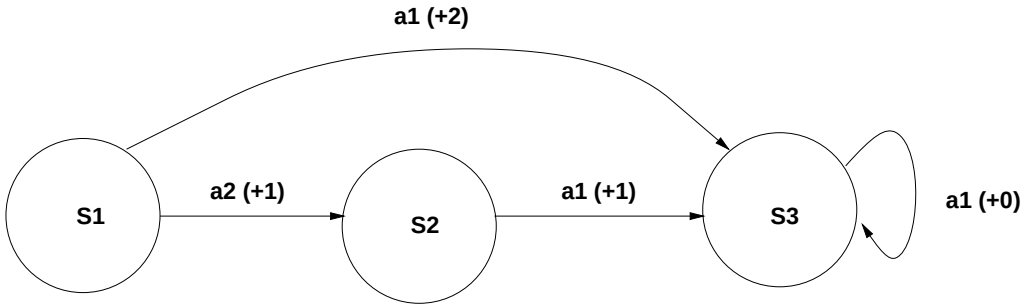

Figure 1: A simple MDP that clarifies the meaning of *n-discount-optimality*. There are only 2 possible policies, corresponding to the two actions in state **S1**. Both policies yield an average reward of 0, and are *-1-discount-optimal*. Both policies yield a cumulative reward of 2 in reaching the absorbing goal state **S3**, and are also *0-discount-optimal*. However, only the top policy is *1-discount-optimal*, since $2 > (1 + \gamma)$, for all $0 \leq \gamma < 1$. Finally, the top policy is also Blackwell optimal, since it is *n-discount-optimal*, for all $n \geq -1$.

4 Sensitive Discount Optimality Equations

In this section we present a series of nested equations that need to be solved in order to compute n-discount-optimal policies, and in particular, Blackwell optimal policies. These equations can be derived from the Laurent series expansion given in Equation 1 of the cumulative discounted reward, although the details of the derivation is beyond the scope of this paper (see [14]).

Let D denote the set of deterministic stationary policies for any finite MDP. The following system of inductively defined equations has to be solved to determine the coefficients of Laurent series expansion of the cumulative discounted reward [14].

$$\max_{\pi \in D_{-2}} (P^\pi - I)\, y_{-1} = 0. \tag{2}$$

$$\max_{\pi \in D_{-1}(y_{-1})} (r_\pi - y_{-1} + (P^\pi - I) y_0) = 0. \tag{3}$$

$$\max_{\pi \in D_n(y_{-1},\ldots,y_n)} (-y_n + (P^\pi - I) y_{n+1}) = 0, \quad \forall n \geq 0. \tag{4}$$

For conciseness, the equations have been written in vector notation, and each equation above actually represents a set of $|S|$ equations, one for each state. The sets D_n refer to nested sets of policies, where $D_m \subseteq D_n$, if $m \geq n$, as illustrated in Figure 2. The base set $D_{-2} = D$ is the set of all deterministic stationary policies. The set $D_{-1}(y_{-1})$ refers to all gain-optimal policies, i.e. all policies that satisfy the first two equations above. Note that the coefficient y_{-1} is simply the average reward. Similarly, the set $D_0(y_{-1}, y_0)$ refers to all bias-optimal policies that satisfy the first two equations, and the third equation for $n = 0$.

Puterman [14] and Veinott [23] have developed provably convergent policy iteration procedures for solving the above nested set of equations for any desired n. In particular, their algorithms can be used to obtain a Blackwell optimal policy, since it can be shown that an

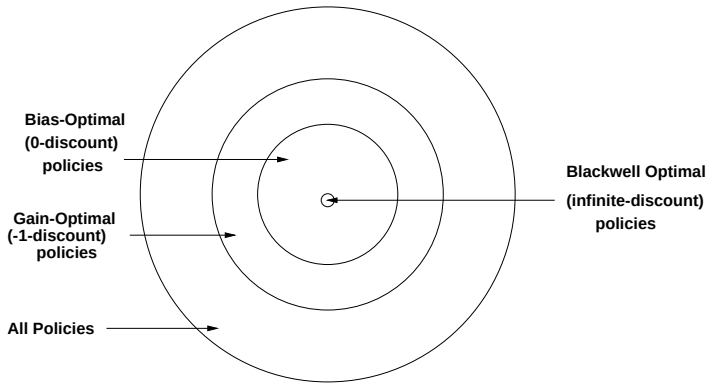

Figure 2: This diagram illustrates the concept of *n-discount-optimal* policies. Gain-optimal policies are *-1-discount-optimal* policies. Bias-optimal (or T-optimal) policies are *0-discount-optimal* policies. Note that *n-discount-optimal* policies get increasingly selective as n increases. Blackwell optimal policies are *n-discount-optimal* for all $n \geq -1$.

$|S| - 2$-discount-optimal policy is Blackwell optimal [14, 23]. In practice, if the set D_n for any n has only one policy, then it is automatically Blackwell optimal, and the algorithm can be terminated. For example, in Figure 1, since D_1 has only one policy, it is Blackwell optimal.

5 Bias Optimality For Unichain MDP's

To illustrate the applicability of the sensitive discount optimality framework to average reward RL, we will develop a novel model-free average reward method for computing bias-optimal (0-discount-optimal) policies. We will assume for simplicity that the MDP is unichain. An MDP is *unichain* if the transition matrix P^π corresponding to every policy π contains a single recurrent class of states, and a (possibly empty) set of transient states. A state is recurrent under a policy if it is visited forever. A transient state under a policy is one that will be visited only a finite number of times. In Figure 1, state **s3** is recurrent and the other states are transient under both policies.

For unichain MDP's, Equation 2 need not be solved since the average reward of any policy is state independent. That is,

$$\rho^\pi(x) = \rho^\pi(y) = \rho^\pi, \quad \forall x, y \in S, \forall \pi.$$

The reason is that unichain policies generate a single recurrent class of states, and possibly a set of transient states. Since states in the recurrent class will be visited forever, the expected average reward cannot differ across these states. Since the transient states will eventually never be reentered, they can at most accumulate a finite total expected reward before entering a recurrent state, which vanishes under the limit. Many interesting problems involve unichain MDP's, such as stochastic grid-world problems [19], admission control queueing systems [14],

and an AGV sorting parts on conveyer belts [21]. To compute bias-optimal (or 0-discount-optimal) policies for a unichain MDP requires solving Equation 3 and Equation 4. These equations are restated in the following two theorems.

Theorem 4 *For any finite unichain MDP, there exists a scalar ρ^* and a value function V^* that satisfy the equation*

$$V^*(x) + \rho^* = \max_{a \in A(x)} \left(r(x,a) + \sum_y P_{xy}(a) V^*(y) \right), \quad \forall x \in S, \tag{5}$$

where the "greedy" policy π^* formed by selecting actions that maximize the right-hand side of the above equation is gain-optimal *(-1-discount-optimal).*

Equation 5 is the well-known average reward Bellman equation, and can be seen to follow directly from Equation 3. All the previous average reward RL algorithms [15, 16, 21] are based on solving this equation, either using a model-based approach or a model-free approach. Solving this equation, however, is not sufficient to produce bias-optimal (0-discount-optimal) policies, even for unichain MDP's [12, 14]. Equation 4 also needs to be solved for this purpose.

Theorem 5 *Let V be a value function, and ρ be a scalar that together satisfy Equation 5. Define $A_V(x) \subseteq A(x)$ to be the set of actions that maximize the right-hand side of Equation 5. There exists a function $W : S \to \mathcal{R}$ satisfying the equation*

$$W(x) = \max_{a \in A_V(x)} \left(-V(x) + \sum_y P_{xy}(a) W(y) \right), \quad \forall x \in S, \tag{6}$$

such that any policy formed by choosing actions in A_V that maximize the right-hand side of the above equation is bias-optimal (0-discount-optimal).

Equation 6 can be seen to follow directly from Equation 4, for $n = 0$. The function W, which we will informally refer to as the *bias offset*, is more precisely the third term y_1 in the Laurent series expansion of the expected cumulative discounted reward given by Equation 1. Note that these optimality equations are nested, since the set A_V of actions over which the maximization is sought in Equation 6 is restricted to those that maximize the right-hand side of Equation 5. The function W holds the key to policy improvement. In particular, let π be a policy for which there exists a bias offset function W satisfying the equation

$$W(x) = \left(-V^\pi(x) + \sum_y P_{xy}(\pi(x)) W(y) \right), \quad \forall x \in S \tag{7}$$

then if there exists a policy $\pi' \neq \pi$ such that

$$\sum_y P_{xy}(\pi'(x)) W(y) \geq \sum_y P_{xy}(\pi(x)) W(y)$$

over all states x, with strict inequality in at least one state, then it follows that

$$V^{\pi'}(x) \geq V^\pi(x), \quad \forall x \in S$$

with strict inequality in at least one state, making π' a better policy than π. Thus, the bias offset function W indicates how close a policy is to achieving bias-optimality. The algorithm described below uses a successive approximation method to estimate both the bias values V and the bias offset values W.

6 A Model-Free Bias Optimality Algorithm

We now present a model-free algorithm for computing bias-optimal (0-discount-optimal) policies. This algorithm can be viewed as extending existing Q-learning type methods for the average reward framework, such as those proposed by Schwartz [15], Singh [16], and Bertsekas [1], to produce bias-optimal policies.

Since the two equations (Equation 5 and Equation 6) are nested, one possibility is to successively approximate the gain ρ and bias values V by solving the first equation, and then solve the second equation using the just computed bias and gain. However, this approach is not guaranteed to converge, since stopping the successive approximation process for computing V values at any point will result in some finite error, which could prevent the solution to the second optimality equation. Instead, we will follow the approach proposed by Federgruen and Schweitzer [4] for solving nested functional equations, and *interleave* the successive approximation process for solving both the equations.

Model-free methods represent the value function using *action-values*. Let us define the action value $R^\pi(x,a)$ to represent the average adjusted value of the non-stationary policy of doing an action a in state x once, and then following policy π subsequently. That is,

$$R^\pi(x,a) = r(x,a) - \rho^\pi + \sum_y P_{xy}(a)V^\pi(y), \tag{8}$$

where the value of state y is defined as

$$V^\pi(y) = \max_{a \in A(y)} R^\pi(y,a),$$

and ρ^π is the average reward of policy π. We can also define an action-value representation of the bias-offset function $W(x)$ as follows:

$$W^\pi(x,a) = -R^\pi(x,a) + \sum_y P_{xy}(a)W^\pi(y),$$

where once again, the bias offset $W(x)$ at any state x is defined using action-values as

$$W^\pi(x) = \max_{a \in A_{R^\pi}(x)} W^\pi(x,a).$$

Note that the maximization above is only with respect to actions that maximize the right-hand side of Equation 8 (denoted above by $A_{R^\pi}(x)$). Since the action values $R^\pi(x,a)$ are themselves being estimated over time, in the algorithm, this maximization is implemented by computing all actions that are within ϵ of $A_{R^\pi}(x)$, where $\epsilon < 1$ is slowly decayed to 0

The model-free algorithm is described in Figure 3. The action values $R^\pi(x,a)$ and $W^\pi(x,a)$, and the average reward ρ^π, are estimated using a relaxation process, controlled by the learning rates β_n, δ_n, and α_n, respectively, all of which are decayed slowly to 0. The action values are set to 0 for a reference state s, which serves to keep the values bounded.

ALGORITHM:

1. Let time step $n = 0$. Initialize $R_n(x,a) = 0$, and $W_n(x,a) = 0$ for all $x \in S$ and $a \in A(x)$. Let the current state be x. Let s be a reference state, preferably one recurrent under all policies. The update rules below fix $R_n(s,i)$ and $W_n(s,i)$ to 0, for some action i. Let $A(x,\epsilon_n)$ represent the set of actions that are within ϵ_n of $\max_{a \in A(x)} R_n(x,a)$.

2. With probability $1-p_{exp}$, select action a to be some $a_o \in A(x,\epsilon_n)$ that maximizes $W_n(x,a_o)$. Otherwise, let action a be any random action $a_r \in A(x)$.

3. Carry out action a. Let the next state be y, and the reward be r_{imm}. If either $x \neq s$ or $a \neq i$, update the R values using the following rule:

$$R_{n+1}(x,a) \leftarrow R_n(x,a)(1-\beta_n) + \beta_n(r_{imm} - \rho_n + \max_{a \in A(y)} R_n(y,a))$$

4. If a non-random action $a = a_o$ was chosen, update the average reward ρ using the rule:

$$\rho_{n+1} \leftarrow \rho_n(1-\alpha_n) + \alpha_n r_{imm}$$

5. If a non-random action $a = a_o$ was chosen, and if either $x \neq s$ or $a \neq i$, update the W values as follows:

$$W_{n+1}(x,a) \leftarrow W_n(x,a)(1-\delta_n) + \delta_n(-R_n(x,a) + \max_{a \in A(y,\epsilon_n)} W_n(y,a))$$

6. If $n <$ MAX_ITERATIONS, set $n \leftarrow n+1$, and the current state x to y and go to step 2.

7. Output $\pi(x) \in \text{argmax}_{a \in A(x,\epsilon_n)} W_n(x,a)$ for all $x \in S$.

Figure 3: A model-free algorithm for computing bias-optimal policies for unichain MDP's.

7 Admission Control Queueing System

In this section we compare the new bias-optimal model-free algorithm with R-learning on a simple admission control queueing system. We have recently undertaken a fairly detailed study of this domain using a model-based bias optimality algorithm [10]. Here, we wish

to simply use this domain to illustrate that the model-free bias optimality algorithm can outperform previous gain-optimal algorithms like R-learning, by being able to learn bias-optimal policies. The admission control system is based on the classic queuing model [8]. 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 of the admission control queuing system 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. Recently, Haviv and Puterman [5] have shown 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 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.

The M/M/1 queuing model is a continuous time MDP, since job arrivals and service times are real valued exponentially distributed random variables. A standard approach called *uniformization* is used to convert the continuous time problem into a discrete time MDP (see [1, 14] for a description of uniformization). Figure 4 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. As described before, λ and μ are the arrival and service times, respectively. 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. 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.

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.

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

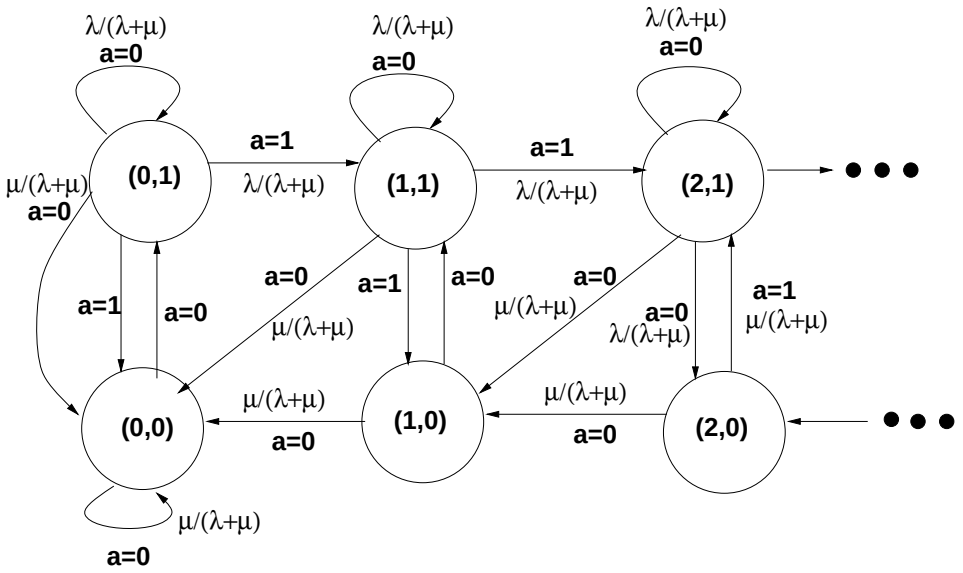

Figure 4: This diagram illustrates the MDP representation of the uniformized M/M/1 admission control queuing system for the average reward case. Each transition is labeled by the action ($a = 0$ or $a = 1$) and its probability of occurrence (e.g. $\frac{\lambda}{\lambda+\mu}$). Not all labels are shown, for clarity. The MDP structure repeats to the right up to the maximum queue size. λ is the arrival rate, and μ is the service rate. $a = 1$ means accept new arrival, and $a = 0$ means reject the new arrival (if there is one), or continue the process.

$$r((s,0),0) \quad = \quad r((s,1),0) = -f(s)(\lambda+\mu), \quad s \geq 1.$$

The two optimality equations will not be shown, but they follow straightforwardly from Equation 5 and Equation 6 and the structure of the MDP in Figure 4. Figure 5 compares the performance of the model-free algorithm (described in Figure 3 above) with R-learning. State $(0,0)$ was chosen as the reference state since it is recurrent under all policies (with action $i = 0$). As the left-hand graph in Figure 5 shows, the average reward of both algorithms is virtually identical, since both of them learn gain-optimal policies. However, as the right-hand graph shows, the new bias optimality algorithm results in much better utilization of the queue, since on the average the queue length is much higher than that for the gain-optimal algorithm. The reason is because the new algorithm infers the bias-optimal admit-$L+1$ policy. Note that the peaks in the queue length graph correspond to the initial transient period when the policy is being learned.

8 Future Work

To our knowledge, this paper represents the first use of sensitive discount optimality to derive an RL algorithm, and raises many interesting issues for further work.

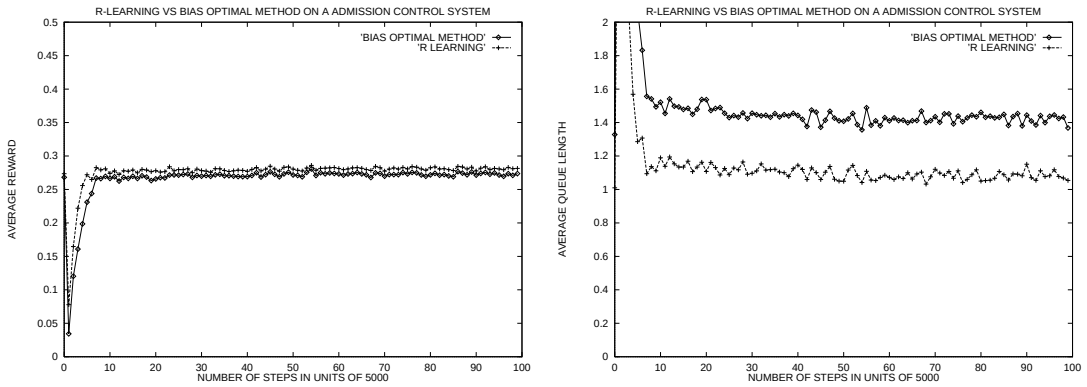

Figure 5: This figure compares the performance of the bias-optimal model-free algorithm with R-learning. The left-hand side graph shows that both algorithms learn gain-optimal policies. However, the right-hand side graph shows that using the bias-optimal algorithm results in much better utilization of the queue, since almost 50% more jobs are serviced on the average. The graphs are averaged over 30 runs. The parameters used in this experiment were $\lambda = 5, \mu = 5, R = 12, c = 1$.

- *Model-based Bias Optimality Algorithm:* The algorithm described in this paper is a model-free algorithm. We have also developed a model-based bias optimality algorithm [10], which extends previous model-based ARL algorithms, such as H-learning [21] and Jalali and Ferguson's B algorithm [6].

- *Sensitive Discount Optimality:* In this paper, we have only focused on the first two terms of sensitive discount optimality, namely gain and bias optimality. One important issue for further research is to provide RL algorithms for further terms in this series, as well as to understand the practical necessity for considering terms beyond the second bias-optimality term.

- *Theoretical Convergence:* A key question is whether the proposed algorithm can be guaranteed to yield a bias-optimal policy. We note that our algorithm uses a successive approximation method for solving nested equations, similar to those that have been theoretically studied in the OR literature [4].

- *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, routing control [9], and robotics related tasks [13].

References

[1] D. Bertsekas. *Dynamic Programming and Optimal Control.* Athena Scientific, 1995.

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

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

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

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

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

[7] L. Kaelbing, editor. *Reinforcement Learning*. Kluwer Academic Press, 1996. Special Issue of Machine Learning Journal, Vol 22.

[8] L. Kleinrock. *Queueing Systems*. John Wiley, 1976.

[9] M. Littman and J. Boyan. Packet routing in dynamically changing networks: A reinforcement learning approach. In *Neural Information Processing Systems (NIPS)*, pages 671–678, 1993.

[10] S. Mahadevan. An average reward reinforcement learning algorithm for computing bias-optimal policies. Submitted.

[11] S. Mahadevan. 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*, pages 164–172. Morgan Kaufmann, 1994.

[12] S. Mahadevan. Average reward reinforcement learning: Foundations, algorithms, and empirical results. *Machine Learning*, 1996. To Appear in Special Issue on Reinforcement Learning (edited by Leslie Kaelbing).

[13] S. Mahadevan and J. Connell. Automatic programming of behavior-based robots using reinforcement learning. *Artificial Intelligence*, 55:311–365, 1992. Appeared originally as IBM TR RC16359, Dec 1990.

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

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

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

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

[18] R. Sutton. Learning to predict by the method of temporal differences. *Machine Learning*, 3:9–44, 1988.

[19] R. Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the Seventh International Conference on Machine Learning*, pages 216–224. Morgan Kaufmann, 1990.

[20] R. Sutton, editor. *Reinforcement Learning*. Kluwer Academic Press, 1992. Special Issue of Machine Learning Journal Vol 8, Nos 3-4, May 1992.

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

[22] J. Tsitsiklis. Asynchronous stochastic approximation and Q-learning. *Machine Learning*, 16:185–202, 1994.

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

[24] C. Watkins. *Learning from Delayed Rewards*. PhD thesis, King's College, Cambridge, England, 1989.