Proceedings of the 2001 Winter Simulation Conference
B. A. Peters, J. S. Smith, D. J. Medeiros, and M. W. Rohrer, eds.

ON IMPROVING THE PERFORMANCE OF SIMULATION-BASED ALGORITHMS FOR AVERAGE REWARD PROCESSES WITH APPLICATION TO NETWORK PRICING

Enrique Campos-Náñez
Stephen D. Patek

Department of Systems and Information Engineering
University of Virginia
Charlottesville, VA 22903, U.S.A.

ABSTRACT

We address performance issues associated with simulation-based algorithms for optimizing Markov reward processes. Specifically, we are concerned with algorithms that exploit the regenerative structure of the process in estimating the gradient of the objective function with the respect to control parameters. In many applications, states which initially have short expected return-times may eventually become infrequently visited as the control parameters are updated. As a result, unbiased updates to the control parameters can become so infrequent as to render the algorithm impractical. The performance of these algorithms can be significantly improved by adapting the state which is used to mark regenerative cycles. In this paper, we introduce such an adaptation procedure, give initial arguments for its convergence properties, and illustrate its application in two numerical examples. The examples relate to the optimal pricing of communication network resources for congestion-controlled traffic.

1 INTRODUCTION

In this paper we address performance issues associated with simulation-based algorithms for optimizing Markov reward processes. We focus on algorithms that apply to recurrent processes which operate by (1) deriving a "noisy" estimate of the gradient of average reward each regeneration cycle and (2) updating control parameters accordingly. An important practical issue associated with this type of algorithm is that regenerative cycles can be extremely long, with correspondingly large variances. This often causes the algorithms to converge slowly, to the point of being impractical.

We address this issue by introducing a mechanism for appropriately selecting and adjusting a special "marked" state i^*, which marks the beginning of each regeneration cycle. The main idea is to allow frequent updating of the control parameters by periodically adjusting (adapting) the marked state according to the most recently observed states in the simulation. Thus, the normal operation of the algorithm is interrupted occasionally (especially in situations where the prevailing i^* is unlikely to be visited again soon) and restarted with a new i^* which is likely to have a shorter expected return time. Shorter return times generally have the desireable effect of reducing the variance of the gradient estimate. Unfortunately, this comes at the expense of throwing away the data associated with naturally occurring, long renewal periods. We address potential problems with bias by forcing the intervals between successive i^* updates to grow on average without bound. We show through some heuristic arguments, and numerical examples, the benefits obtained through the adoption of the proposed mechanism.

1.1 Related Work

The concept of optimizing a system through simulation, or even direct observation, has been studied extensively (see Fu and Hu 1997 for an overview of the use of simulation in optimization). Factorial designs, response surfaces, and meta models are some examples of these techniques, each of which seeks to estimate the gradient of a system's performance measure with respect to a continuous control parameter.

In this paper, we are concerned with Monte-Carlo techniques for gradient estimation, and their use in system optimization, through stochastic approximation recursions (see Kushner and Yin 1997). Estimates of the gradient of an objective function can be obtained by simple procedures, for example by obtaining symmetric samples of the desired performance measures around the current value of the control parameters. This is the basic nature of the *Kiefer-Wolfowitz* algorithm (see Kiefer and Wolfowitz 1952). The slow convergence of this method is related to the large number of observations required for each gradient estimate and the fact that these observations are biased.

Perturbation analysis, introduced by Ho in (Ho, Eyler, and Chien 1979), uses small perturbations to the control parameter to obtain unbiased gradient estimates through comparison with the unperturbed system (see also (Ho and Cao

1983), (Suri 1987), and (Suri 1988)). Another approach is frequency domain experimentation Jacobson 1993, in which the control parameters are adjusted according to harmonic (sinusoidal) functions of time. Other methods to obtain unbiased gradient estimates, such as the likelihood ratio method, introduced by Glynn in (Glynn 1986; Glynn 1987), exploit the stochastic structure of the system, can operate within a single simulation run, and can be implemented on-line. The use of unbiased estimates in a stochastic approximation constitutes a *Robbins-Monro*-type algorithm (see Kushner and Yin 1997). Adaptive versions of stochastic approximation algorithms, where system parameters are unknown and need to be tracked during simulation, have also been studied, see (Benveniste, Métivier, and Priouret 1987).

Other simulation based algorithms, not discussed in this paper, attempt to solve Markov decisions processes by estimating value functions explicitly (see Bertsekas and Tsitsiklis 1996). These methods are also referred to as *Critic* algorithms, for the role the value function plays in evaluating the performance of the parameters. *Actor-Critic* methods (see Konda and Tsitsiklis 2001) build value function approximations, and use them to inform a gradient-descent type algorithm.

The techniques presented in this paper apply to the two algorithms presented by Marbach and Tsitsiklis in (Marbach and Tsitsiklis 2001) for optimizing a Markov reward process. Their first algorithm, a "batch" algorithm, constructs unbiased gradient estimates in a fashion similar to (Glynn 1986), based on the information collected during regeneration cycles. As in other gradient estimation algorithms, visits to a particular state mark the end of regenerative cycles at which point a Robbins-Monro recursion is used to update control parameter values. Updates to the control parameters in this algorithm are made once per regeneration period. The second algorithm of Marbach and Tsitsiklis, an "on-line" algorithm, is a modified version of the first where control parameters can be updated at every step in the simulation, often resulting in faster convergence.

The rest of this paper is organized as follows. In Section 2, we review the algorithms proposed in (Marbach and Tsitsiklis 2001) and discuss performance issues relating to lengthy regeneration cycles. In Section 3, we formally introduce a new technique, i^*-adaptation, which addresses these issues. In Section 4, we provide heuristic arguments to establish the convergence of the modified algorithms. In Section 5, we illustrate our approach with two examples that relate to the pricing of resources in a communication network. Finally in Section 6, we present our conclusions and discuss directions for future research.

2 SIMULATION-BASED OPTIMIZATION ALGORITHMS

In this section, we review the two algorithms analyzed by Marbach and Tsitsiklis in (Marbach and Tsitsiklis 2001). These are the simulation-based methods to which we apply our i^*-adaptation method.

We consider a discrete-time Markov reward process with state space S. Let i_n be the state of the system after n state-transitions in the simulation. We will subsequently use $\theta \in \Theta$ to denote the vector of control parameters that modulate the evolution of the process, and we will use $\tilde{\lambda}(\theta)$ as an estimate of the average reward associated with θ. (To simplify notation in the sequel, we will drop the dependence on θ, leaving $\tilde{\lambda}$ as the current estimate of average reward.) We use $g_i(\theta)$ to denote the expected transition cost from $i \in S$ associated with θ. Similarly, we use $P_{ij}(\theta)$ to denote the probability that the system transitions to state j, given that it is currently at state i, again associated with the parameter vector θ. We use i^* to denote a special recurrent state, the "marked" state, which is used by the simulation-based algorithms to mark the beginning of each new renewal period. The following assumptions, taken from (Marbach and Tsitsiklis 2001) are required by the algorithm.

Assumption 1: The Markov chain corresponding to every policy θ is aperiodic. Furthermore, there exists a state i^* which is recurrent for every such Markov chain.

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

Assumption 3: For every i and j, there exists a bounded function $L_{ij}(\theta)$ such that

$$\nabla p_{ij}(\theta) = p_{ij}(\theta) L_{ij}(\theta) \qquad \forall \theta.$$

2.1 "Batch" Algorithm

Let $t_0, t_1, \ldots, t_m, \ldots$ be the times of visits to i^*.

1. After the m-th visit to i^*, simulate state transitions $i_n \to i_{n+1}$ under the parameter vector θ_m, where $n = t_m, t_m + 1, \ldots$.
2. If $i_{n+1} = i^*$, and corresponds to the $m+1$ visit to the marked state ($t_{m+1} = n$), compute the final estimate of the gradient $F_m(\theta_m, \tilde{\lambda}_m)$, defined by

$$\sum_{n=t_m}^{t_{m+1}-1} \left(\tilde{v}_{i_n}(\theta_m, \tilde{\lambda}_m) L_{i_{n-1}, i_n}(\theta_m) + \nabla g_{i_n}(\theta_m) \right), \tag{1}$$

where

$$\tilde{v}_{i_n}(\theta_m, \tilde{\lambda}_m) = \sum_{k=n}^{t_{m+1}-1} \left(g_{i_k}(\theta_k) - \tilde{\lambda}_m\right), \quad (2)$$

and $L_{i_{n-1},i_n}(\theta) = \nabla P_{i_{n-1},i_n}(\theta)/P_{i_{n-1},i_n}(\theta)$. Since $\tilde{v}_{i_n}(\theta_m, \tilde{\lambda}_m)$ is the accumulation of the differences between incurred costs and the average reward, we must have $\tilde{v}_{i^*}(\theta_m, \tilde{\lambda}_m) := 0$.

3. Update the control parameters θ in the direction of the gradient estimate and correct the estimate of the average reward:

$$\begin{aligned} \theta_{m+1} &= \theta_m + \gamma_m F_m(\theta, \tilde{\lambda}_m), \\ \tilde{\lambda}_{m+1} &= \tilde{\lambda}_m + \eta\gamma_m \sum_{n=t_m}^{t_{m+1}-1} \left(g_{i_n}(\theta_m) - \tilde{\lambda}_m\right). \end{aligned}$$

Here γ_m is a decreasing stepsize such that $\sum_{m=0}^{\infty} \gamma_m = \infty$, and $\sum_{m=0}^{\infty} \gamma_m^2 < \infty$, and $\eta > 0$.

A convergence result, (Marbach and Tsitsiklis 2001), shows that under assumptions 1-3, and following the stepsize rules mentioned above, $\lambda(\theta_m)$ converges, and $\nabla\lambda(\theta_m) \to 0$ as $m \to \infty$.

2.2 "On-line" Algorithm

The on-line algorithm requires in addition:

Assumption 4: There exists a state $i^* \in S$ and a positive integer N_0, such that, for every state $i \in S$ and every collection $\{P_1, ..., P_{N_0}\}$ of N_0 matrices in the set $\{P(\theta) | \theta \in \Re^K\}$, we have

$$\sum_{n=1}^{N_0} \left[\prod_{l=1}^{n} P_l\right]_{ii^*} > 0.$$

Assumption 5: The stepsizes γ_n are nonincreasing. Furthermore, there exists a positive integer p and a positive scalar A such that

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

The algorithm proceeds by executing the following at each state transition $i_n \to i_{n+1}$ (under the parameter vector θ_n).

1. If $i_{n+1} = i^*$, let $z_n = 0$, otherwise, let

$$z_{n+1} = z_n + L_{i_n,i_{n+1}}(\theta_n). \quad (3)$$

2. Update the control parameter, along with the estimate of average reward:

$$\theta_{n+1} = \theta_n + \gamma_n\left(\nabla g(i_n, \theta_n) + z_n(g(i_n, \theta_n) - \tilde{\lambda}_n)\right), \quad (4)$$

$$\tilde{\lambda}_{n+1} = \tilde{\lambda}_n + \eta\gamma_n\left(g(i_n, \theta_n) - \tilde{\lambda}_n\right). \quad (5)$$

Another result from (Marbach and Tsitsiklis 2001), shows that under assumptions 1-5, $\lambda(\theta_n)$ converges, and $\nabla\lambda(\theta_n) \to 0$ as $n \to \infty$.

2.3 Performance of the Simulation-based Algorithms

In both the batch and on-line versions of the simulation-based method above, the marked state i^* is chosen in advance and is held constant throughout the course of the simulation. Although i^* is theoretically recurrent for all θ, changes to the control parameters during the simulation can introduce extremely long return-times. This is particularly true in queueing-type applications (as in Section 5), where a natural choice for i^* is the "system empty" state. As the parameter vector evolves, i.e. as the system becomes more heavily loaded, the "system empty" state becomes relatively infrequent.

The existence of long regenerative cycles has a negative impact on both of the simulation algorithms above. On the one hand, when transitions to i^* are infrequent, the "batch" algorithm has few opportunities to update the control parameters, and, when updates do occur, the variance of the gradient estimate is correspondingly large. This leads to extremely slow convergence in practical applications. On the other hand, when transitions to i^* are infrequent, the on-line algorithm has few opportunities to correct biased observations, leading to apparent instability, despite theoretical convergence guarantees.

3 ADJUSTING THE MARKED STATE

To address the practical issue of long regeneration cycles, we introduce a new mechanism, i^*-adaptation, which seeks to periodically adjust the marked state in accordance with the states most recently observed in the simulation. The goal is to allow more frequent updates with less variance in the gradient estimate, without introducing bias (at least over the long term).

3.1 Rules for Selecting i^*

We propose a simple procedure: if the length of the current regenerative cycle exceeds a given threshold, we interrupt the process (without stopping the simulation) to select a new state value for i^* and, after appropriately reinitializing the data structures, we resume the gradient estimation and parameter updates. Before formally describing our i^* update procedure, we make the following simplifying assumption.

Assumption 1b. P_θ, the stochastic matrix of the process under control policy θ, is irreducible under all control parameters θ.

Without this assumption, it would be possible to choose new state values for i^* which are not recurrent under the prevailing value of θ (or which quickly become non-recurrent as the parameter vector evolves).

The i^* adaptation process begins with an initial marked state $i_0^* \in S$, a threshold $\tau_0 > 0$ which limits the length of a regenerative cycle, and $t_0 = 0$ to mark the start time. Subsequently, we let

$$t_m = \min\left\{t_{m-1} + \tau_{m-1}, \min\{t > t_{m-1} | i_t = i^*_{m-1}\}\right\},$$

for all $m \geq 1$. Thus, t_m marks the time when either a regenerative cycle has been completed or abandoned. We simulate the process, and, at stage t_m, we employ the following procedure to select a new state value for i^*.

Generic Update Rule If the current regenerative cycle has exceeded its limits ($i_{t_m} \neq i^*_{m-1}$), we select the current state as the marked state. This is,

$$i^*_m = \begin{cases} i_{t_m} & \text{if } i_{t_m} \neq i^*_{m-1}, \\ i^*_{m-1} & \text{otherwise.} \end{cases}$$

In problems with the structure of a generalized birth-death processes (as in Section 5), it is possible to use an alternative update rule. Let us assume that each state i of the system can be expressed as an N-tuple, $i \leftrightarrow x^i = (x^i_1, x^i_2, \ldots, x^i_N)$, where each x^i_k has a quantitative interpretation such as the "the number of class k customers associated with state i." With this representation in hand, it is possible to define a long-run "average" state

$$\bar{x}_m = \left(\frac{1}{t_m}\sum_{t=0}^{t_m} x_1(t), \ldots, \frac{1}{t_m}\sum_{t=0}^{t_m} x_N(t)\right),$$

where $x_k(t)$ is the k-th component of the N-tuple associated with state of the system at stage t.

GBD Update Rule We select a new marked state by moving in the direction of the average observed state. Specifically, we set

$$i^*_m = \begin{cases} \lfloor a i^*_{m-1} + (1-a)\bar{i}_m \rfloor & \text{if } i_{t_m} = i^*_{m-1}, \\ i^*_{m-1} & \text{otherwise.} \end{cases}$$

Here $0 < a < 1$. In the examples of Section 5, we set $a = 1/2$.

By throwing away regenerative cycles of length greater than the threshold τ_m we introduce bias into both the gradient estimate and the estimate of average reward. To overcome this problem, we increase the algorithm's tolerance to lengthy cycles every time that a cycle is dropped. To do this we let

$$\tau_{m+1} = \begin{cases} \tau_m & \text{if } i_{t_m+1} = i^*_m, \\ \tau_m(1+b) & \text{otherwise.} \end{cases}$$

for some $b > 0$. Numerical results suggest that any positive constant b eliminates the bias introduced by discarding some of the observed cycles.

3.2 Batch Algorithm with i^*-Adaptation

Let $\{m_n\}_{n=0}^{\infty}$ be the subsequence of cycles such that $i_{t_{m_n}} = i^*_{m_n-1}$; which we will call "successful" cycles. The batch algorithm with i^*-adaptation consists of updating the parameter values and the average reward estimate according to

$$\begin{aligned} \theta_{n+1} &= \theta_n + \gamma_n F_{m_n}\left(\theta_n, \tilde{\lambda}_n\right), \\ \tilde{\lambda}_{n+1} &= \tilde{\lambda}_n + \eta\gamma_n \sum_{k=t_{m_n}}^{t_{m_{n+1}-1}} \left(g_{i_k}(\theta_n) - \tilde{\lambda}_n\right). \end{aligned}$$

where $F_{m_n}(\theta_n, \tilde{\lambda}_n)$ is the gradient estimate obtained during the successful cycle m_n, using control θ_n, and average reward estimate $\tilde{\lambda}_n$ and calculated according to (1).

3.3 On-line Algorithm with i^*-Adaptation

The on-line algorithm with i^* adaptation operates as follows. At each simulation step k, we update the values of θ_{k+1}, $\tilde{\lambda}_{k+1}$, and z_{k+1}, according to (3)-(5). If $k + 1 = t_m$, we let $z_{k+1} = 0$. Additionally, if the cycle was unsuccessful, we recover the parameter values and average reward estimations from the beginning of the last cycle; that is

$$\begin{aligned} \theta_{k+1} &= \theta_{t_{m-1}}, \\ \tilde{\lambda}_{k+1} &= \tilde{\lambda}_{t_{m-1}}. \end{aligned}$$

This way we effectively drop any changes to the parameters during an unsuccessful cycle, and recover the original algorithm. This requires that the algorithm to store $K + 1$ additional parameters, where K is the dimension of the parameter vector θ.

4 CONVERGENCE ANALYSIS

To prove that θ_m converges to a local minimum using either the batch or on-line algorithms with i^* adaptation, it is necessary to show that the bias introduced by dropping unsuccessful cycles disappears as $m \to \infty$. In this section we provide some considerations for the batch algorithm that show some of the issues involved.

As proved in (Marbach and Tsitsiklis 2001), there exist $C > 0$, and $\rho \in (0,1)$ such that for all states $k \in S$ and parameter vectors θ,

$$P_\theta\{T = k\} \leq C\rho^k,$$

where T is the number of transitions starting from k and ending at the first return to k. This allows us to derive a useful bound on the expected bias introduced at each parameter update. Let

$$f(\theta_m, \tilde{\lambda}_m) = E_\theta\left[F_m(\theta_m, \tilde{\lambda}_m)\right],$$

be the expected value of the gradient estimate, under the original batch algorithm. Let $s_m = \min\{t \geq t_{m-1} \mid i_t = i^*\}$, and let

$$\hat{f}(\theta_m, \tilde{\lambda}_m) = E_\theta\left[F_m(\theta_m, \tilde{\lambda}_m)\,|s_m - t_{m-1} \leq \tau_m\right],$$

be the biased expected gradient estimate given that the cycle is successful. (Here, the conditional expectation of the gradient ignores the possibility that we can have a regeneration cycle of length greater than τ_m.) Since

$$P_\theta\{s_m - t_{m-1} > \tau_m\} \leq C\frac{\rho^{\tau_m}}{1-\rho},$$

we have that

$$||\hat{f}(\theta_m, \tilde{\lambda}_m) - f(\theta_m, \tilde{\lambda}_m)|| \leq C_F C \frac{\rho^{\tau_m}}{1-\rho},$$

where C_F is a bound on $f(\theta, \lambda)$ for all θ, λ (cf. Marbach and Tsitsiklis 2001). Thus, the bias in the estimate of the gradient necessarily converges to zero as $\tau_m \to \infty$. A similar argument, using the fact that $\tilde{v}_i(\theta, \tilde{\lambda})$ in (2) is bounded for all states i (cf.Marbach and Tsitsiklis 2001), shows that the updates to the estimate of average reward $\tilde{\lambda}_m$ eventually become consistent with the updates in the original method.

5 APPLICATIONS TO PRICING OF NETWORK SERVICES

We consider two numerical examples below that relate to pricing mechanisms for splitting the capacity of a congested

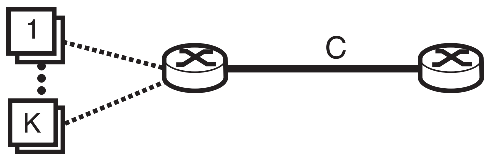

Figure 1: Depiction of the Examples Considered in this Paper

network resource among the competing users. We adopt a network-centered view, where the service provider acts as a central control authority, fixing prices for access to the resource to optimize a specific objective, e.g. the expected rate at which revenue accrues.

Our examples relate to the generic scenario depicted in Figure 1, where a single link is shared by users belonging to different classes, with diverse bandwidth requirements and demand patterns. We assume that average rate at which requests for file transfers (calls) arrive depends on the vector of prices u set for each call type. (Thus, u will play the role of the control parameters θ in the simulation based algorithms of Sections 2 and 3). We assume that price information can be instantaneously communicated to the users, much in the spirit of (Paschalidis and Tsitsiklis 2000), and (Patek and Campos-Nanez 2000). Regulating access to a network resource through prices can also be seen as a soft or probabilistic form of admission control.

Further, we are interested in solving this problem for heterogeneous traffic, where calls belong to either a *guaranteed-rate* class (where each call requires a fixed amount of capacity) or a *best-effort* class (where calls receive an adjustable portion of the capacity not consumed by the guaranteed-rate calls).

Scenarios that involve best-effort traffic tend to exhibit heavy congestion, since an increasing number of best-effort calls has a corresponding decrease in the capacity allocated to them. As a consequence, the "system empty" state, which is commonly used to mark regenerative cycles in queueing applications, is infrequently visited and is therefore an impractical choice for i^*. In fact, in examples of this type, the best choice for i^* is strongly dependent on the prevailing set of prices, and there is a real need for i^*-adaptation in simulation-based algorithms.

5.1 Formulation

Consider a setting where a single link with capacity C is shared by K classes of traffic. (In the examples of the following subsections, we have $K = 1, 2$.) We assume that call arrivals and departures are exponentially distributed, so $i = (i_1, \ldots, i_K)$ denotes the state of the system, where i_k is the number of class-k calls currently in the system. New class-k calls arrive at instantaneous rates, denoted $\alpha_k(u_k)$, regulated by the price u_k set for the class. We use β_k to denote the instantaneous average rate at which class-k calls depart the system. We use $K_g \subset K$ to denote the subset of guaranteed-rate flows. For each class $k \in K_g$, we use b_k to denote the bandwidth requirement for this class, and the departure rate β_k for this class does not depend on the state of the system.

For each best effort class $k \in K - K_g$, the rate at which calls depart the system does depend on the state of the system, as follows. We assume that the average size of a file transfer is B_k so that the instantaneous departure rate is $\beta_k(i) = \frac{b_k(i)}{B_k}$, where

$$b_k(i) = \min\left\{ M, \frac{C - \sum_{j \in K_g} i_j b_j}{\sum_{j \in K - K_g} i_j} \right\}.$$

Thus, expected service times increase as a function of number of best-effort users in the system. The equal sharing of unreserved capacity in our model corresponds to the assumption that a congestion-control mechanism is in place, such as TCP.

5.1.1 Objective Function

Our objective in both examples below is to maximize the average rate at which revenue accrues. That is, we seek to maximize, through the choice of u in the set U of admissible prices, the expression

$$J_u(i) = \lim_{T \to \infty} \frac{1}{T} E\left\{ \int_0^T \sum_{k=0}^{K} \alpha_k(u_k) u_k dt \,\middle|\, i(0) = i \right\} \quad (6)$$

for all $i \in S$. Since it is assumed that all states are recurrent, we have that $J_u(i) = J_u(j) = \lambda(u)$ for all $i, j \in S$.

Alternatively, we could have formulated this problem as a large-scale continuous-time Markov decision process, where we would have sought to optimize the mapping from $i \in S$ to admissible prices U. Interestingly, in many pricing contexts, the gain in revenue associated with dynamic pricing is relatively small compared to the complexity of solving (and later implementing) the optimization problem. (Patek and Campos-Nanez 2000). In fact, for the case $K_g = K$, Paschalidis and Tsitsiklis (Paschalidis and Tsitsiklis 2000) have shown that in the many-small-users limit optimal static prices are nearly optimal within the full set class of dynamic pricing policies.

5.2 Example 1: N On-Off Sources

In this example we consider a single class of $N = 100$ best-effort users that share a single link through access lines of capacity $M = 56.6$ kbps. The users switch between being "on," where they are in the process of downloading a file, and being "off," where they are waiting for the next opportunity to initiate a download. (On/off models of this type are common for capturing the basic nature of web-traffic.) We assume that the average rate $\phi(u)$ at which an individual user switches from "off" to "on" depends on the price $u \in (0, 1)$ as $\phi(u) = \phi_0 \cdot (1 - u)$. Thus, $\alpha_k(u) = \phi_0(N - i)(1 - u)$ is the rate at which new calls arrive to the system, where i is the current number of "on" users. See Table 1 for additional example parameters.

Table 1: Parameters for Example 1

Best-Effort	
C	1.544 Mbps
M	56.6 Kbps
B	64 KB
N	100 Users
ϕ_0	.1 calls per sec.

5.2.1 Performance of the Algorithms

After uniformizing the process, we employed the simulation-based algorithms from (Marbach and Tsitsiklis 2001), with and without the i^*-adaptation procedure from Section 3, to find a value of u that maximizes average reward. The results are plotted in Figures 2 and 3, for the batch and on-line versions of the algorithm, respectively. We note that in all cases we are able to identify the revenue maximizing price u^*. Although not shown in the plot, the pure batch and on-line algorithms work *only* when the fixed value of i^* is carefully chosen and the system is initiazed with a nearly optimal solution. In general, the i^*-adaptation (using wither update rule) serves to reduce the variance in the estimate of the optimal price and at the same time improves the robustness of the scheme, allowing us to start with any arbitrarily selected i^* and initial price estimate.

5.3 Example 2: Heterogeneous Traffic

In this example, two classes of traffic are considered, one class (class-1) with guaranteed rate requirements, the other (class-2) being a best-effort class downloading files of average length $B = 64$ kB. The average rate $\alpha_1(u_1)$ of call arrivals for the guaranteed-rate class is given by $\bar{\alpha}_1 \cdot (1 - u_1/4)$, with $u_1 \in (0, 4)$. The average rate $\alpha_2(u_2)$ of call arrivals for the best effort class is given by $\bar{\alpha}_2 \cdot (1 - u_2)$, with $u_2 \in (0, 1)$. See Table 2 for additional information on the parameter values for this example.

5.3.1 Performance of the Algorithms

Again, after uniformizing the process, we employed the simulation-based algorithms from (Marbach and Tsitsiklis 2001) to optimize u_1 and u_2, with and without the i^*-

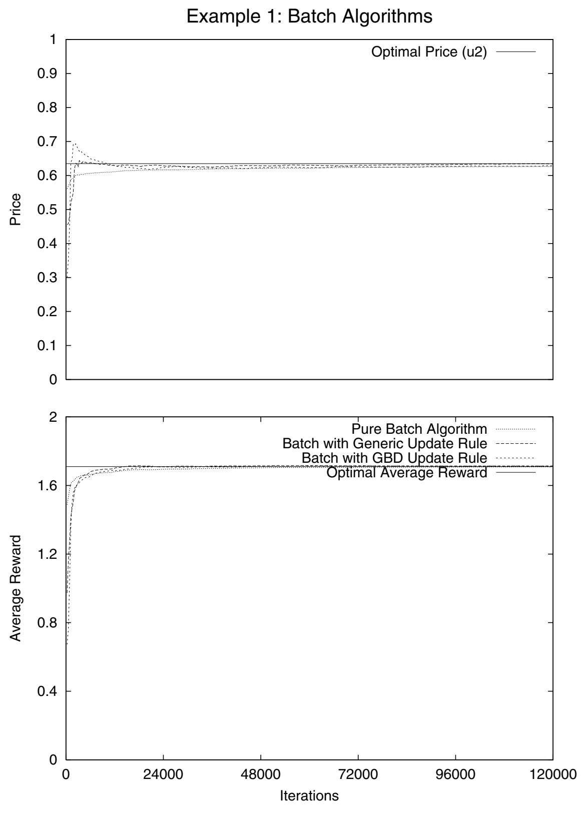

Figure 2: Performance of the Batch Algorithm for Example 1

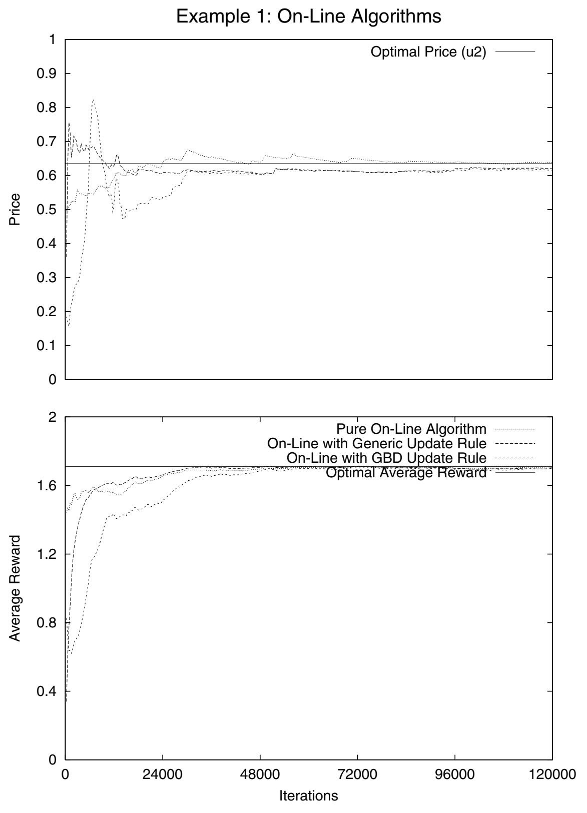

Figure 3: Performance of the On-Line Algorithm for Example 1

Table 2: Parameters for Example 2

C	1.544 Mbps	
	Guaranteed-Rate, $k = 1$	**Best-Effort,** $k = 2$
M_k	128 Kbps	56.6Kbps (max)
β_k	1/180	$\frac{b_2(i)}{B}$
$\bar{\alpha}_k$	1/5	1/2

adaptation procedure from Section 3. Interestingly, we were unable to get the pure versions of the batch and on-line algorithms to converge to a reasonable solution. We tried many different (fixed) values for i^*, and we tried initializing the price estimates very close to the optimal solution, and still the pure versions of the simulation-based algorithms failed to yield meaningful results. On the other hand, *with* i^*-adaptation (using either update rule), the simulation based methods work remarkably well. The results are shown in Figures 4 and 5, respectively.

6 CONCLUSIONS

We have introduced a new method, i^*-adaptation, for improving the performance of simulation-based optimization algorithms. While a thorough analysis of the method is currently in progress, we have outlined some of the main theoretical issues in proving the convergence of the procedure. Our numerical results indicate that i^*-adaptation introduces no significant bias in the long term, while significantly improving the robustness of the underlying method. Recall that, in Example 1, in order to get the pure versions of the batch and on-line algorithms to converge, we had to cheat by initializing the price estimate very close to the optimal solution. In Example 2, no amount of cheating helped. We conclude that i^*-adaptation is a useful device for improving the performance of simulation-based optimization algorithms for average reward processes.

ACKNOWLEDGMENTS

This work was supported in part by National Science Foundation Grants, ECS-9875688 (CAREER) and ANI-9903001, and by scholarship 68428 of Consejo Nacional de Ciencia y Tecnología.

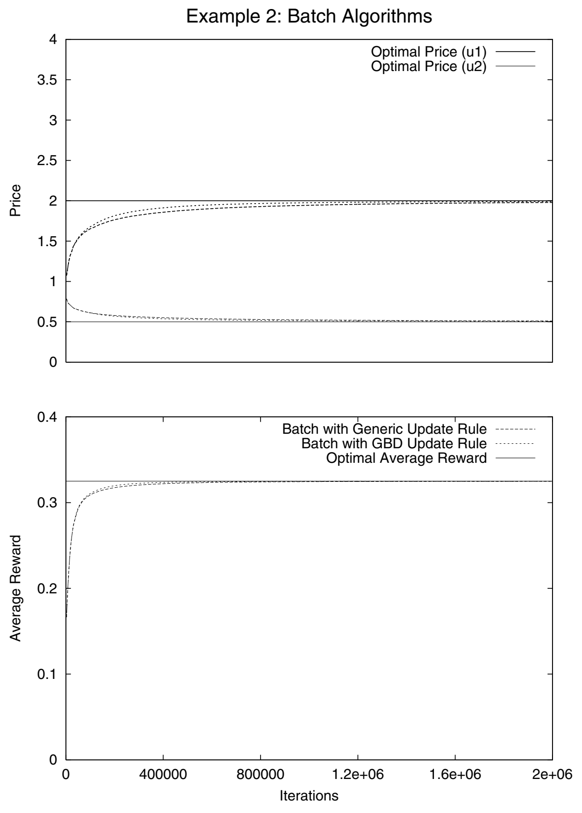

Figure 4: Performance of the Batch Algorithm for Example 2

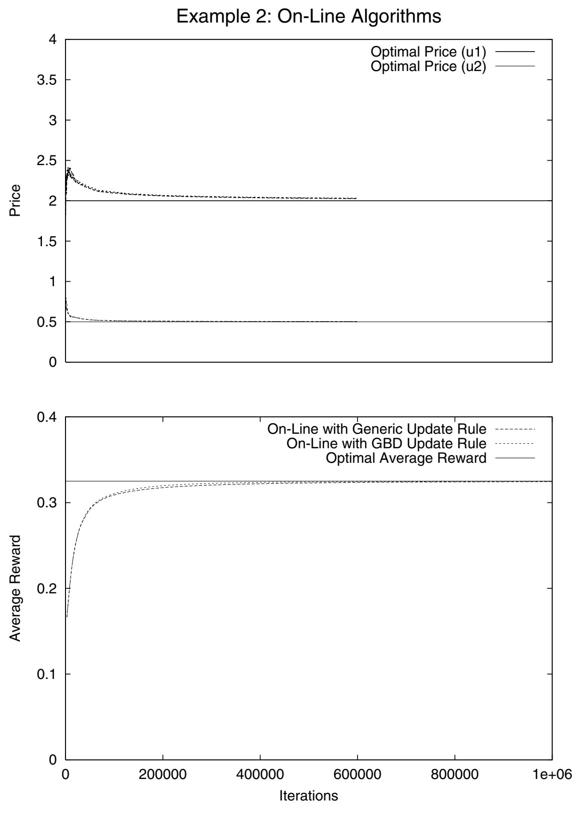

Figure 5: Performance of the On-Line Algorithm for Example 2

REFERENCES

Benveniste, A., M. Métivier, and P. Priouret. 1987. *Adaptive algorithms and stochastic approximations*. Springer-Verlag.

Bertsekas, D., and J. S. Tsitsiklis. 1996. *Neuro-dynamic programming*. Athena Scientific.

Fu, M., and J. Q. Hu. 1997. *Conditional monte carlo: Gradient estimation and optimization applications*. Kluwer Academic Publishers.

Glynn, P. W. 1986. Stochastic approximation for monte carlo optimization. *Proceedings of the 1986 Winter Simulation Conference*.

Glynn, P. W. 1987. Likelihood ratio gradient estimation: An overview. *Proceedings of the 1997 Winter Simulation Conference*.

Ho, Y. C., and X. R. Cao. 1983. Perturbation analysis and optimization of queueing networks. *J. of Optimization Theory and Applications* 40:559–582.

Ho, Y. C., M. A. Eyler, and T. T. Chien. 1979. A gradient technique for general buffer storage design in a serial production line. *J. Prod. Res.* 17:557–580.

Jacobson, S. H. 1993. Variance and bias reduction techniques for harmonic gradient estimators. *Applied Mathematics and Computation*:153–186.

Kiefer, J., and J. Wolfowitz. 1952. Stochastic estimation of the maximum of a regression function. *Annals of Mathematical Statistics* 23:462–466.

Konda, W. R., and J. N. Tsitsiklis. 2001. Actor-critic algorithms. *to appear*.

Kushner, H. J., and G. G. Yin. 1997. *Stochastic approximation algorithms and applications*. Springer-Verlag.

Marbach, P., and J. N. Tsitsiklis. 2001. Simulation-based optimization of markov reward processes. *to appear in ACM Transaction on Automatic Control*.

Paschalidis, I. C., and J. N. Tsitsiklis. 2000. Congestion-dependent pricing of network services. *IEEE/ACM Transactions on Networking*.

Patek, S. D., and E. Campos-Nanez. 2000. Pricing of dialup services: an example of congestion-dependent pricing in the internet. *IEEE Conference on Decision and Control CDC 2000*.

Suri, R. 1987. Infinitesimal analysis for general discrete event systems. *J. Assoc. Comput. Mach.* 34:686–717.

Suri, R. 1988. Perturbation analysis gives strongly consistent estimates for the m/g/1 queue. *Management Science* 34:39–64.

AUTHOR BIOGRAPHIES

ENRIQUE CAMPOS-NÁÑEZ is Ph.D. Student at the Department of Systems and Information Engineering at the University of Virginia. His research interest are in the use simulation based optimization techniques, and adaptive control, to implement pricing and other mechanisms to regulate the use of network resources. He received his Bachelor's degree in Mathematics from Universidad Nacional Autónoma de México, in 1990, a M. Sc. in Operations Research from Stanford University, in 1996. His email address is *ec3z@virginia.edu*.

STEPHEN D. PATEK is an Assistant Professor in the Department of Systems and Information Engineering at the University of Virginia. His research interests are in Markov decision processes and stochastic control, with applications to data networks and traffic engineering. He received his Bachelor's degree in Electrical Engineering in 1991 from the University of Tennessee, Knoxville. He received his Ph.D. in Electrical Engineering and Computer Science in 1997 from the Massachusetts Institute of Technology. His email address is *patek@virginia.edu*.