A Discrete Parameter Stochastic Approximation Algorithm for Simulation Optimization

Shalabh Bhatnagar
Department of Computer Science and Automation
Indian Institute of Science
Bangalore 560 012, India
shalabh@csa.iisc.ernet.in

Hemant J. Kowshik
Department of Electrical Engineering
Indian Institute of Technology Madras
Chennai 600 036, India

The authors develop a two-timescale simultaneous perturbation stochastic approximation algorithm for simulation-based parameter optimization over discrete sets. This algorithm is applicable in cases where the cost to be optimized is in itself the long-run average of certain cost functions whose noisy estimates are obtained via simulation. The authors present the convergence analysis of their algorithm. Next, they study applications of their algorithm to the problem of admission control in communication networks. They study this problem under two different experimental settings and consider appropriate continuous time queuing models in both settings. Their algorithm finds optimal threshold-type policies within suitable parameterized classes of these. They show results of several experiments for different network parameters and rejection cost. The authors also study the sensitivity of their algorithm with respect to its parameters and step sizes. The results obtained are along expected lines.

Keywords: Discrete parameter optimization, stochastic approximation algorithms, two-timescale SPSA, admission control in communication networks

1. Introduction

Stochastic discrete optimization plays an important role in the design and analysis of discrete event systems. Examples include the problem of resource (buffer/bandwidth) allocation in manufacturing systems and communication networks [1, 2], as well as admission control and routing in communication/wireless networks [3]. Discrete optimization problems in general are hard combinatorial problems.

There have been several approaches for solving discrete optimization problems. Among these, simulated annealing [4] and its variants have been well studied. Here the algorithm does not necessarily proceed along the path of decreasing cost as increases in cost function are allowed with a certain probability. The original annealing algorithm, however, requires cost function measurements to be precisely known. Various variants of the above algorithm have subsequently been developed that work with noisy or imprecise measurements [5] or use simulation [6]. Alrefaei and Andradóttir [6] also propose the use of a "constant temperature" annealing schedule instead of a (slowly) decreasing schedule. Some other variants of simulated annealing that work with noisy objective function estimates include the stochastic ruler algorithm of Yan and Mukai [7] and the stochastic comparison algorithm of Gong, Ho, and Zhai [8]. Algorithms based on simulated annealing, however, are known to be slow in general. In some other work, a stochastic branch and bound algorithm for problems of discrete stochastic optimization, subject to constraints that are possibly given by a set of inequalities, has been developed in Norkin,

SIMULATION, Vol. 81, Issue 11, November 2005 757-772

DOI: 10.1177/0037549705062294

Ermoliev, and Ruszczyński [9]. In Barnhart, Wieselthier, and Ephremides [3], the problem of admission control in multihop wireless networks is considered, and a gradient search-based procedure is used for finding an optimal policy within the class of certain coordinate convex policies; see also Jordan and Varaiya [10], who consider optimization on the above type of policies for an admission control problem in multiple-service, multiple-resource networks.

In Gokbayrak and Cassandras [2], a stochastic discrete resource allocation problem is considered. The discrete optimization problem is transformed into an analogous (surrogate) continuous parameter optimization problem by constructing the convex hull of the discrete search space. An estimate of the cost derivative is obtained using concurrent estimation [11] or perturbation analysis (PA) [12] techniques. For the surrogate problem, they use the simplex method to identify the $(N + 1)$ points in the discrete constraint set whose convex combination the $\mathcal{R}^N$-valued surrogate state is. In Gokbayrak and Cassandras [13], more general constraint sets than the one above are considered, although the basic approach is similar (to Gokbayrak and Cassandras [2]). A computationally simpler algorithm (than the simplex method) is provided for identifying the above-mentioned points. (The basic approach in Gokbayrak and Cassandras [2, 13] is somewhat loosely similar to our approach below. We provide detailed comparisons in the next section.)

In the works cited above, noisy estimates of the cost function are assumed available at discrete parameter settings, with the goal being to find the optimum parameter for the noise-averaged cost. There are, however, scenarios in which the cost for a given parameter value is in itself the long-run average of a certain cost function whose noisy estimates at the (above) given parameter values can be obtained via simulation. For solving such problems, there have been approaches in the continuous optimization framework. For instance, those based on PA [12] or the likelihood ratio [14] require one simulation for finding the optimum parameter. These, however, require knowledge of sample path derivatives of performance with respect to (w.r.t.) the given parameter. In addition, they require certain constraining regularity conditions on sample performance and underlying process parameters. Among finite difference gradient approximation-based approaches, the Kiefer-Wolfowitz (K-W) algorithm with two-sided differences requires $2N$ loss function measurements for an N-dimensional parameter vector. In a recent related work [15], the gradient estimates in a K-W-type algorithm are chosen by simultaneously perturbing all parameter components along random directions, most commonly by using independent, symmetric, ± 1-valued, Bernoulli-distributed, random variables. This algorithm, known as the simultaneous perturbation stochastic approximation (SPSA) algorithm, requires only two loss function measurements for any N-vector parameter and is, in general, found to be very efficient.

In Bhatnagar and Borkar [16], a two-timescale stochastic approximation algorithm that uses one-sided difference K-W-type gradient estimates was developed as an alternative to PA-type schemes. The idea here is that aggregation/averaging of data is performed using the faster timescale recursions while parameter updates are performed on the slower one, and the entire algorithm is updated at each epoch. The disadvantage with this algorithm, however, is that it requires a significant amount of computation since it generates $(N + 1)$ parallel simulations at each instant and hence is slow when N is large. In Bhatnagar et al. [17], the SPSA-based analog of the algorithm in Bhatnagar and Borkar [16] was developed, except for the difference that an averaging of the faster timescale recursions over a certain fixed number $(L \geq 1)$ of epochs was proposed before each parameter update, in addition to the two-timescale averaging. This number L is set arbitrarily, and the additional averaging is seen to improve performance. Other simulation optimization algorithms based on the SPSA technique have more recently been developed in Bhatnagar and Borkar [18], Bhatnagar et al. [19], and Bhatnagar [20].

The algorithms described in the previous two paragraphs are for optimization over continuously valued sets. In this article, we develop an algorithm for discrete parameter simulation-based optimization where the cost is the long-run average of certain cost functions that depend on the state of an underlying parameterized Markov process. This algorithm is a variant of the algorithm in Bhatnagar et al. [17] that is adapted to optimization over discrete sets and uses two-timescale averaging. The motivation for using two-timescale stochastic approximation is shown in the next section. In a related work [1], a variant of the SPSA algorithm [15] is used for function optimization over discrete sets. This, however, is of the one-timescale variety and is not developed for the setting of simulation optimization as in this study. We briefly explain the convergence analysis of our algorithm. Next, we present an application of our algorithm for finding optimal policies for a problem of admission control [21, 22] in communication networks. Our methods are applicable to both admission control of calls (e.g., of customers who require services with certain quality of service [QoS] requirements for an arbitrary or random duration of time) as well as packet admission control (or that for individual packets at link routers within the network). We consider two different settings for our experiments. These are explained in detail in section 4. We assume the class of feedback policies in these settings to be of the threshold type that depend on the state of the underlying Markov process at any instant. Our algorithm gives an optimal policy within the prescribed class of policies (i.e., computes the optimal

such threshold-type policy). Obtaining optimal policies using analytical techniques in such settings is not feasible in general. Moreover, as we explain in the next section, PA-type approaches, as with Gokbayrak and Cassandras [2, 13], are not directly applicable on such settings. Our simulation-based discrete parameter algorithm proves effective here as it gives an optimal policy within a given parameterized class of policies (in this case, the threshold-type policies). This is similar in spirit to neurodynamic programming techniques with policy parameterization in the context of Markov decision processes [23, 24].

The rest of the article is organized as follows: section 2 describes the setting and the algorithm and provides a motivation for the two-timescale idea. We also discuss comparisons of our approach here with the ones in Gokbayrak and Cassandras [2, 13]. The convergence analysis is briefly presented in section 3. The experiments on admission control in communication networks are presented in section 4. Finally, section 5 provides the concluding remarks.

2. Framework and Algorithm

Consider a Markov process $\{X_n^\theta\}$ parameterized by $\theta \in D \subset \mathcal{Z}^N$, where $\mathcal{Z}$ is the set of all integers and $N \geq 1$. Suppose X_n^θ, $n \geq 1$, take values in $S \subset \mathcal{R}^d$ for some $d \geq 1$. We assume D has the form $D = \prod_{i=1}^N \{D_{i,\min}, \ldots, D_{i,\max}\}$. Here, $D_{i,\min}, D_{i,\max} \in \mathcal{Z}$ with $D_{i,\min} \leq D_{i,\max}$, $i = 1, \ldots, N$, and $\{D_{i,\min}, \ldots, D_{i,\max}\}$ is the set of successive integers from $D_{i,\min}$ to $D_{i,\max}$, with both end points included. Assume that for any fixed $\theta \in D$, $\{X_n^\theta\}$ is ergodic as well with transition kernel $p_\theta(x, dy)$, $x, y \in S$. Let $h : \mathcal{R}^d \to \mathcal{R}$ be the associated cost function that is assumed to be bounded and continuous. The aim is to find $\theta^* \in D$ such that

$$J(\theta^*) = \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} h(X_i^{\theta^*}) = \min_{\theta\in D} J(\theta). \qquad (1)$$

Here, $J(\cdot)$ denotes the long-run average cost. The aim therefore is to find θ^* that minimizes $J(\theta)$.

Before we proceed further, we first provide the motivation for using the two-timescale stochastic approximation approach. Suppose for the moment that we are interested in finding a parameter θ, taking values in a continuously valued set, that minimizes $J(\theta)$. One then needs to compute the gradient $\nabla J(\theta) \equiv (\nabla_1 J(\theta), \ldots, \nabla_N J(\theta))^T$, assuming that it exists. Consider a Markov chain $\{X_j^\theta\}$ governed by parameter θ. Also, consider N other Markov chains $\{X_j^i\}$ governed by parameters $\theta + \delta e_i$, $i = 1, \ldots, N$, respectively. Here, e_i is an N-dimensional vector with 1 in the ith place and 0 elsewhere. Then,

$$\nabla_i J(\theta) = \lim_{\delta\to 0} \frac{(J(\theta + \delta e_i) - J(\theta))}{\delta} \qquad (2)$$

$$= \lim_{\delta\to 0} \left(\lim_{l\to\infty} \frac{1}{\delta l} \sum_{j=0}^{l-1} (h(X_j^i) - h(X_j^\theta)) \right). \qquad (3)$$

The gradient $\nabla J(\theta)$ may thus be estimated by simulating the outcomes of the $(N + 1)$ Markov chains $\{X_j^\theta\}$ and $\{X_j^i\}$, $i = 1, \ldots, N$, respectively. These Markov chains, in turn, correspond to the underlying processes in independently running parallel systems that are each identical to one another except that they run with different parameter values (above). Note that the transition dynamics of these Markov chains need not be precisely known as long as state transitions of these chains can be simulated. Estimates (2) correspond to one-sided Kiefer-Wolfowitz estimates of the gradient $\nabla J(\theta)$. One can similarly obtain two-sided Kiefer-Wolfowitz gradient estimates as well. More recently, Spall [15] proposed the following (SPSA) randomized difference gradient estimates. Suppose $\triangle_i$, $i = 1, \ldots, N$, are independent identically distributed (i.i.d.) Bernoulli-distributed random variables with $\triangle_i = \pm 1$ w.p. 1/2, and let $\triangle = (\triangle_1, \ldots, \triangle_N)^T$. Quantities $\triangle$ are called random perturbations. (More general conditions on the sequences of independently generated $\triangle$s are given in Spall [15]; see also condition (A) below, which is similar to Spall [15].) Consider now Markov chains $\{X_n^+\}$ and $\{X_n^-\}$ that are respectively governed by parameters $(\theta + \delta\triangle)$ and $(\theta - \delta\triangle)$. Then, the SPSA gradient estimate is

$$\nabla_i J(\theta) = \lim_{\delta\to 0} E\left[\frac{(J(\theta + \delta\triangle) - J(\theta - \delta\triangle))}{2\delta\triangle_i} \right] \qquad (4)$$

$$= \lim_{\delta\to 0} E\left[\lim_{l\to\infty} \frac{1}{2\delta l} \sum_{j=0}^{l-1} (h(X_j^+) - h(X_j^-)) \right], \qquad (5)$$

where the expectation is taken w.r.t. the common distribution of $\triangle_i$.

Note from the form (3) of the estimates above, it is clear that the outer limit is taken only after the inner limit. For classes of systems for which the two limits (in (3)) can be interchanged, gradient estimates from a single sample path can be obtained using PA [12] or likelihood ratio [14] techniques. For a general class of systems for which the above limit interchange is not permissible, any recursive algorithm that computes optimum θ should have two loops, with the outer loop (corresponding to parameter updates) being updated once after the inner loop (corresponding to data averages), for a given parameter

value, has converged. Thus, one can physically identify separate timescales: the faster one on which data based on a fixed parameter value are aggregated and averaged and the slower one on which the parameter is updated once the averaging is complete for one latter iteration. The same effect as on different physical timescales can also be achieved by using different step-size schedules (also called timescales) in the stochastic approximation algorithm. Note also from the form of (5) that the expectation is taken after the inner loop limit and is followed by the outer limit. Bhatnagar and Borkar [16] develop a two-timescale algorithm using (3) as the gradient estimate, while Bhatnagar et al. [17] do so with (5). The use of the latter (SPSA) estimate was found in Bhatnagar et al. [17] to significantly improve performance in two-timescale algorithms. As explained before, the algorithms of Bhatnagar and Borkar [16] and Bhatnagar et al. [17] were for the setting of continuous parameter optimization.

In what follows, we develop a variant of the algorithm in Bhatnagar et al. [17] for discrete parameter stochastic optimization. Consider two step size schedules or timescale sequences $\{a(n)\}$ and $\{b(n)\}$ that satisfy $a(n), b(n) > 0\ \forall n$, and

$$\sum_n a(n) = \sum_n b(n) = \infty,$$

$$\sum_n (a(n)^2 + b(n)^2) < \infty,$$

$$a(n) = o(b(n)). \tag{6}$$

Let $c > 0$ be a given constant. Also, let $\Gamma_i(x)$ denote the projection from $x \in \mathcal{R}$ to $\{D_{i,\min}, \ldots, D_{i,\max}\}$. In other words, if the point x is an integer and lies within the set $\{D_{i,\min}, \ldots, D_{i,\max}\}$, then $\Gamma_i(x) = x$, else $\Gamma_i(x)$ is the closest integer within the above set to x. Furthermore, if x is equidistant from two points in $\{D_{i,\min}, \ldots, D_{i,\max}\}$, then $\Gamma_i(x)$ is arbitrarily set to one of them. For $y = (y_1, \ldots y_N)^T \in \mathcal{R}^N$, suppose $\Gamma(y) = (\Gamma_1(y_1), \ldots, \Gamma_N(y_N))^T$. Then, $\Gamma(y)$ denotes the projection of $y \in \mathcal{R}^N$ on the set D (defined earlier).

Now suppose for any $n \geq 0$, $\triangle(n) \in \mathcal{R}^N$ is a vector of mutually independent and mean zero random variables $\{\triangle_1(n), \ldots, \triangle_N(n)\}$ (namely, $\triangle(n) \stackrel{\triangle}{=} (\triangle_1(n), \ldots, \triangle_N(n))^T$), taking values in a compact set $E \subset \mathcal{R}^N$ and having a common distribution. We assume that random variables $\triangle_i(n)$ satisfy condition (A) below.

Condition (A). There exists a constant $\bar{K} < \infty$, such that for any $n \geq 0$, and $i \in \{1, \ldots, N\}$, $E\left[(\triangle_i(n))^{-2}\right] \leq \bar{K}$.

We also assume that $\triangle(n)$, $n \geq 0$, are mutually independent vectors and that $\triangle(n)$ is independent of the σ-field $\sigma(\theta(l), l \leq n)$. Condition (A) is a somewhat standard requirement in SPSA algorithms; see, for instance, Spall [15] for similar conditions. Let F denote the common distribution function of the random variables $\triangle_i(n)$.

2.1 Discrete Parameter Stochastic Approximation Algorithm

- Step 0 (Initialize): Set $Z^1(0) = Z^2(0) = 0$. Fix $\theta_1(0)$, $\theta_2(0)$, $\ldots$, $\theta_N(0)$ within the set D and form the parameter vector $\theta(0) \stackrel{\triangle}{=} (\theta_1(0), \ldots, \theta_N(0))^T$. Fix L and (large) M arbitrarily. Set $n := 0$, $m := 0$ and fix a constant $c > 0$. Generate i.i.d. random variables $\triangle_1(0), \triangle_2(0), \ldots, \triangle_N(0)$, each with distribution F and such that these are independent of $\theta(0)$. Set $\theta_j^1(0) \stackrel{\triangle}{=} \Gamma_j(\theta_j(0) - c\triangle_j(0))$ and $\theta_j^2(0) \stackrel{\triangle}{=} \Gamma_j(\theta_j(0) + c\triangle_j(0))$, $j = 1, \ldots, N$, respectively.
- Step 1: Generate simulations $X_{nL+m}^{\theta^1(n)}$ and $X_{nL+m}^{\theta^2(n)}$, respectively governed by $\theta^1(n) \stackrel{\triangle}{=} (\theta_1^1(n), \ldots, \theta_N^1(n))^T$ and $\theta^2(n) \stackrel{\triangle}{=} (\theta_1^2(n), \ldots, \theta_N^2(n))^T$. Next, update

$$Z^1(nL+m+1) = Z^1(nL+m) + b(n)(h(X_{nL+m}^{\theta^1(n)}) - Z^1(nL+m)),$$

$$Z^2(nL+m+1) = Z^2(nL+m) + b(n)(h(X_{nL+m}^{\theta^2(n)}) - Z^2(nL+m)).$$

  If $m = L-1$, set $nL := nL + L$, $m := 0$ and go to step 2;

  else, set $m := m + 1$ and repeat step 1.
- Step 2: For $i = 1, \ldots, N$,

$$\theta_i(n+1) = \Gamma_i\left(\theta_i(n) + a(n)\left[\frac{Z^1(nL) - Z^2(nL)}{2c\triangle_i(n)}\right]\right).$$

  Set $n := n + 1$. If $n = M$, go to step 3;

  else, generate i.i.d. random variables $\triangle_1(n), \triangle_2(n), \ldots, \triangle_N(n)$ with distribution F (independent of their previous values and also of previous and current θ values). Set $\theta_j^1(n) := \Gamma_j(\theta_j(n) - c\triangle_j(n))$, $\theta_j^2(n) := \Gamma_j(\theta_j(n) + c\triangle_j(n))$, $j = 1, \ldots, N$, and go to step 1.
- Step 3 (termination): Terminate algorithm and output $\theta(M) \stackrel{\triangle}{=} (\theta_1(M), \ldots, \theta_N(M))^T$ as the final parameter vector.

In the above, $Z^1(nL+m)$ and $Z^2(nL+m)$, $m = 0, 1, \ldots, L-1$, $n \geq 0$, are defined according to their corresponding recursions in step 1 and are used for averaging the cost function in the algorithm. As stated earlier,

we use only two simulations here for any N-vector parameter. Note that in the above, we also allow for an additional averaging over L (possibly greater than 1) epochs in the two simulations. This is seen to improve performance of the system, particularly when the parameter dimension N is large. We study system performance with varying values of L in our experiments. Note also that the value of c should be carefully chosen here. In particular, $c > 1/(2R)$ should be used, with R being the maximum absolute value that the random variables $\triangle_j(n)$ can take. This is because for values of c that are lower, both perturbed parameters $\theta_j^1(n)$ and $\theta_j^2(n)$ (after being projected onto the grid) would always lead to the same point. The algorithm would not show good convergence behavior in such a case. For instance, when $\triangle_j(n)$ are i.i.d. symmetric, Bernoulli-distributed, random variables with $\triangle_j(n) = \pm 1$ w.p. 1/2, the value of c chosen should be greater than 0.5. This fact is verified through our experiments as well, where we generate perturbations according to the above distribution. Moreover, one should choose appropriate step size sequences that do not decrease too fast so as to allow for better exploration of the search space.

We now provide some comparisons with the approach used in Gokbayrak and Cassandras [2, 13] as the basic approach used in the above references is somewhat loosely similar to ours. The method in Gokbayrak and Cassandras [2, 13] uses a (surrogate) system with parameters in a continuously valued set that is obtained as a convex hull of the underlying discrete set. The algorithm used there updates parameters in the above set, using a continuous optimization algorithm by considering the sample cost for the surrogate system to be a continuously valued linear interpolation of costs for the original system parameters. The surrogate parameters after each update are projected onto the original discrete set to obtain the corresponding parameter value for the original system. The algorithm itself uses PA/concurrent estimation-based sample gradient estimates, with the operating parameter being the corresponding projected "discrete" update. Note that there are crucial differences between our approach and the one above. As shown in the next section, we also use a convex hull of the discrete parameter set, but this is only done for proving the convergence of our algorithm that updates parameters directly in the discrete set. We do not require the continuous surrogate system at all during the optimization process. Furthermore, since we use an SPSA-based finite difference gradient estimate, we do not need PA-type gradient estimates. We directly work (for the gradient estimates) with the projected perturbed parameter values in the discrete set. As stated earlier, the use of PA requires constraining conditions on the cost and system parameters so as to allow for an interchange between the expectation and gradient (of cost) operators. We do not require such an interchange because of the use of two timescales, unlike Gokbayrak and Cassandras [2, 13], who use only one. In the type of cost functions and underlying process parameters used in our experiments in section 4, PA is not directly applicable. Finally, Gokbayrak and Cassandras [2, 13] require that the "neighborhood" $(N + 1)$ points, whose convex combination any $\mathcal{R}^N$-valued state of the surrogate system is, need to be explicitly obtained, which is computationally cumbersome. Our algorithm does not require such an identification of points as it directly updates the algorithm on the discrete set. Next, we present the convergence analysis for our algorithm.

3. Convergence Analysis

Suppose D^c denotes the convex hull of the set D. Then D^c has the form $\prod_{i=1}^{N}[D_{i,\min}, D_{i,\max}]$—namely, it is the Cartesian product of the intervals $[D_{i,\min}, D_{i,\max}] \subset \mathcal{R}$, $i = 1, \ldots, N$. For showing convergence, we first extend the dynamics of the Markov process $\{X_n^\theta\}$ to the continuously valued parameter set $\theta \in D^c$ and show the analysis for the extended parameter process. Convergence for the original process is then straightforward. Note that the set D contains only a finite number (say, m) of points. For simplicity, we enumerate these as $\theta^1, \ldots, \theta^m$. Thus, $D = \{\theta^1, \ldots, \theta^m\}$. The set D^c then corresponds to

$$D^c = \{\sum_{i=1}^{m} \alpha_i \theta^i \;\mid\; \alpha_i \geq 0,\ i = 1, \ldots, m,\ \sum_{i=1}^{m} \alpha_i = 1\}.$$

In general, any point $\theta \in D^c$ can be written as $\theta = \sum_{i=1}^{m} \alpha_i(\theta)\theta^i$. Here we consider an explicit functional dependence of the weights α_i, $i = 1, \ldots, m$, on the parameter θ and assume that $\alpha_i(\theta)$, $i = 1, \ldots, m$, are continuously differentiable with respect to θ. For $\theta \in D^c$, define the transition kernel $p_\theta(x, dy)$, $x, y \in S$ as

$$p_\theta(x, dy) = \sum_{k=1}^{m} \alpha_k(\theta) p_{\theta^k}(x, dy).$$

It is easy to verify that $p_\theta(x, dy)$, $\theta \in D^c$, satisfy properties of a transition kernel and that $p_\theta(x, dy)$ are continuously differentiable with respect to θ. Moreover, it is easy to verify that the Markov process $\{X_n^\theta\}$ continues to be ergodic for every $\theta \in D^c$ as well. Let $J^c(\theta)$ denote the long-run average cost corresponding to $\theta \in D^c$. Then, $J^c(\theta) = J(\theta)$ for $\theta \in D$. For a set A, suppose $|A|$ denotes its cardinality. The following lemma is valid for

finite state Markov chains, that is, where $|S| < \infty$. We have

LEMMA 1. For $|S| < \infty$, $J^c(\theta)$ is continuously differentiable in θ.

Proof. Note that $J^c(\theta)$ here can be written as

$$J^c(\theta) = \sum_{i \in S} h(i)\mu_i(\theta),$$

where $\mu_i(\theta)$ is the steady-state probability of the chain $\{X_n^\theta\}$ being in state $i \in S$ for given $\theta \in D^c$. Then it is sufficient to show that $\mu_i(\theta)$, $i \in S$ are continuously differentiable functions. Let for fixed θ, $P(\theta) := [[p_\theta(i, j)]]_{i,j \in S}$ denote the transition probability matrix of the Markov chain $\{X_n^\theta\}$, and let $\mu(\theta) := [\mu_i(\theta)]_{i \in S}$ denote the vector of stationary probabilities. Here we denote by $p_\theta(i, j)$ the transition probabilities of this chain. Also let $Z(\theta) := [I - P(\theta) - P^\infty(\theta)]^{-1}$, where I is the identity matrix and $P^\infty(\theta) = \lim_{m \to \infty}(P(\theta) + \ldots + P^m(\theta))/m$. Here, $P^m(\theta)$ corresponds to the m-step transition probability matrix with elements $p_\theta^m(i, j) = Pr(X_m^\theta = j \mid X_0^\theta = i)$, $i, j \in S$. For simplicity, let us first consider θ to be a scalar. Then, from theorem 2 of Schweitzer [25, pp. 402-3], one can write

$$\mu(\theta + h) = \mu(\theta)(I + (P(\theta + h) - P(\theta))Z(\theta) + o(h)). \tag{7}$$

Thus,

$$\mu'(\theta) = \mu(\theta)P'(\theta)Z(\theta).$$

Hence, $\mu'(\theta)$ (the derivative of $\mu(\theta)$) exists if $P'(\theta)$ (the derivative of $P(\theta)$) does. Note that by construction, for $\theta \in D^c$, $P'(\theta)$ exists and is continuous. Next, note that

$$\begin{aligned}
|\mu'(\theta + h) - \mu'(\theta)| \leq\ & |\mu(\theta + h)P'(\theta + h)Z(\theta + h) \\
& - \mu(\theta)P'(\theta + h)Z(\theta + h)| \\
& + |\mu(\theta)P'(\theta + h)Z(\theta + h) \\
& - \mu(\theta)P'(\theta)Z(\theta + h)| \\
& + |\mu(\theta)P'(\theta)Z(\theta + h) \\
& - \mu(\theta)P'(\theta)Z(\theta)|.
\end{aligned}$$

Then, (again) from theorem 2 [25, pp. 402-3], one can write $Z(\theta + h)$ as

$$\begin{aligned}
Z(\theta + h) = & Z(\theta)H(\theta, \theta + h) \\
& - P^\infty(\theta)H(\theta, \theta + h)U(\theta, \theta + h) \\
& Z(\theta)H(\theta, \theta + h),
\end{aligned}$$

where

$$H(\theta, \theta + h) = [I - (P(\theta + h) - P(\theta))]^{-1} \to I \quad \text{as} \quad |h| \to 0$$

and

$$U(\theta, \theta + h) = (P(\theta + h) - P(\theta))Z(\theta) \to \bar{0} \quad \text{as} \quad |h| \to 0.$$

In the above, $\bar{0}$ is the matrix (of appropriate dimension) with all zero elements. It thus follows that $Z(\theta + h) \to Z(\theta)$ as $|h| \to 0$. Moreover $\mu(\theta)$ is continuous since it is differentiable. Thus, from above, $\mu'(\theta)$ is continuous in θ as well, and the claim follows. For vector θ, a similar proof as above verifies the claim. □

REMARK. Lemma 1 is an important result that is required to push through a Taylor series argument in the following analysis. Moreover, $J^c(\theta)$ turns out to be the associated Liapunov function for the corresponding ordinary differential equation (ODE) (8) that is asymptotically tracked by the algorithm. However, as stated before, lemma 1 is valid only for the case of finite state Markov chains. This is because the result of Schweitzer [25] used in the proof is valid only for finite state chains. For general state Markov processes, certain sufficient conditions for the differentiability of the stationary distribution are given in Vazquez-Abad and Kushner [26]. These are, however, difficult to verify in practice. In the remainder of the analysis (for the case of general state Markov processes taking values in $\mathcal{R}^d$), we simply assume that $J^c(\theta)$ is continuously differentiable in θ. For our numerical experiments in section 4, we require only the setting of a finite state Markov chain, for which lemma 1 holds.

Let us now define another projection function Γ^c: $\mathcal{R}^N \to D^c$ in the same manner as the projection function Γ, except that the new function projects points in $\mathcal{R}^N$ on the set D^c in place of D. Now consider the algorithm described in section 2, with the difference that we use projection $\Gamma_i^c(\cdot)$ in place of $\Gamma_i(\cdot)$ in step 2 of the algorithm, where $\Gamma_i^c(\cdot)$, $i = 1, \ldots, N$ are defined suitably (as with $\Gamma_i(\cdot)$). The above thus corresponds to the continuous parameter version of the algorithm described in section 2. We now describe the analysis for the above (continuous parameter) algorithm with $\Gamma^c(\cdot)$ in place of $\Gamma(\cdot)$. For any bounded and continuous function $v(\cdot) \in \mathcal{R}$, let $\hat{\Gamma}_i^c(\cdot)$, $i = 1, \ldots, N$ be defined by

$$\hat{\Gamma}_i^c(v(x)) = \lim_{\eta \downarrow 0} \left(\frac{\Gamma_i^c(x + \eta v(x)) - x}{\eta} \right).$$

Next for $y = (y_1, \ldots, y_N)^T \in \mathcal{R}^N$, let $\hat{\Gamma}^c(y) = (\hat{\Gamma}_1^c(y_1),$

$\ldots, \hat{\Gamma}_N^c(y_N))^T$. Consider the following ODE:

$$\dot{\theta}(t) = \hat{\Gamma}^c\left(-\nabla J^c(\theta)\right). \tag{8}$$

Let $K \stackrel{\triangle}{=} \{\theta \in D^c \mid \hat{\Gamma}^c\ (\nabla J^c(\theta)) = 0\}$ denote the set of all fixed points of the ODE (8). Suppose for given $\epsilon > 0$, K^ϵ denotes the set $K^\epsilon \stackrel{\triangle}{=} \{\theta \mid \| \theta - \theta' \| \leq \epsilon\ \forall\ \theta' \in K\}$. Then we have the following:

THEOREM 1. Given $\epsilon > 0$, there exists $c_0 > 0$ such that for any $c \in (0, c_0]$, the continuous parameter version of the algorithm converges to some $\theta^* \in K^\epsilon$, as the number of iterations $M \to \infty$.

Proof. The proof proceeds through a series of approximation steps as in Bhatnagar et al. [17]. We briefly sketch it below. Let us define a new sequence $\{\widetilde{b}(n)\}$ of step sizes according to $\widetilde{b}(n) = b\left(\left[\frac{n}{L}\right]\right)$, where $\left[\frac{n}{L}\right]$ denotes the integer part of $\frac{n}{L}$. It is then easy to see that $\{\widetilde{b}(n)\}$ satisfies

$$\sum_n \widetilde{b}(n) = \infty, \quad \sum_n \widetilde{b}(n)^2 < \infty, \quad a(n) = o(\widetilde{b}(n)).$$

In fact, $b(n)$ goes to zero faster than $\widetilde{b}(n)$ does, and thus $\widetilde{b}(n)$ corresponds to an "even faster timescale" than $b(n)$. Now define $\{t(n)\}$ and $\{s(n)\}$ as follows: $t(0) = 0$ and $t(n) = \sum_{m=1}^{n} \widetilde{b}(m)$, $n \geq 1$. Furthermore, $s(0) = 0$ and $s(n) = \sum_{m=1}^{n} a(m)$, $n \geq 1$. Also, let $\triangle(t) = \triangle(n)$ for $t \in [s(n), s(n+1)]$. It is then easy to see that the continuous parameter version of the algorithm above, on this timescale, tracks the trajectories of the system of ODEs:

$$\dot{\theta}(t) = 0, \tag{9}$$

$$\dot{Z}^1(t) = J^c(\theta(t) - c\triangle(t)) - Z^1(t), \tag{10}$$

$$\dot{Z}^2(t) = J^c(\theta(t) + c\triangle(t)) - Z^2(t), \tag{11}$$

in the following manner: suppose we define continuous time processes $\{Y^1(t)\}$ and $\{Y^2(t)\}$ according to $Y^1(t(n)) = Z^1(nL)$, $Y^2(t(n)) = Z^2(nL)$, and for $t \in [t(n), t(n+1)]$, $Y^1(t)$ and $Y^2(t)$ are continuously interpolated from the values they take at the boundaries of these intervals. Furthermore, let us define a real-valued sequence $\{T_n\}$ as follows: suppose $T > 0$ is a given constant. Then, $T_0 = 0$, and for $n \geq 1$,

$$T_n = \min\{t(m)\ \mid\ t(m) \geq T_{n-1} + T\}.$$

Thus $T_n - T_{n-1} \approx T, \forall n \geq 1$. Also, for any n, there exists some integer m_n such that $T_n = t(m_n)$. Now define processes $\{Y^{1,n}(t)\}$ and $\{Y^{2,n}(t)\}$ according to $Y^{1,n}(T_n) = Y^1(t(m_n)) = Z^1(nL)$, $Y^{2,n}(T_n) = Y^2(t(m_n)) = Z^2(nL)$, and for $t \in [T_n, T_{n+1})$, $Y^{1,n}(t)$ and $Y^{2,n}(t)$ evolve according to the ODEs (10)-(11). By an application of Gronwall's inequality, it is easy to see that

$$\sup_{t\in[T_n,T_{n+1})} \| Y^{1,n}(t) - Y^1(t) \|,$$

$$\sup_{t\in[T_n,T_{n+1})} \| Y^{2,n}(t) - Y^2(t) \| \to 0 \text{ as } n \to \infty.$$

Now note that the iteration in step 2 of the continuous version of the algorithm can be written as follows: for $i = 1, \ldots, N$,

$$\theta_i(n+1) = \Gamma_i^c(\theta_i(n) + \widetilde{b}(n)\eta_i(n)),$$

where $\eta_i(n) = o(1)$, $\forall i = 1, \ldots, N$, since $a(n) = o(\widetilde{b}(n))$. Let us now define two continuous time processes $\{\theta(t)\}$ and $\{\hat{\theta}(t)\}$ as follows: $\theta(t(n)) = \theta(n) = (\theta_1(n), \ldots, \theta_N(n))^T$, $n \geq 1$. Also, for $t \in [t(n), t(n+1)]$, $\theta(t)$ is continuously interpolated from the values it takes at the boundaries of these intervals. Furthermore, $\hat{\theta}(T_n) = \hat{\theta}(t(m_n)) = \theta(n)$, and for $t \in [T_n, T_{n+1})$, $\hat{\theta}(t)$ evolves according to the ODE (9). Thus, given $T, \eta > 0$, $\exists P$ such that $\forall n \geq P$, $\sup_{t\in[T_n,T_{n+1})} \| Y^{i,n}(t) - Y^i(t) \| < \eta$, $i = 1, 2$. Also, $\sup_{t\in[T_n,T_{n+1})} \| \hat{\theta}(t) - \theta(t) \| < \eta$. It is now easy to see (cf. [27]) that $\| Z^1(nL) - J^c(\theta(n) - c\triangle(n)) \|$ and $\| Z^2(nL) - J^c(\theta(n) + c\triangle(n)) \| \to 0$ as $n \to \infty$.

Now note that because of the above, the iteration in step 2 of the new algorithm can be written as follows: for $i = 1, \ldots, N$,

$$\theta_i(n+1) = \Gamma_i^c\,(\theta_i(n) + a(n) \left(\frac{J^c(\theta(n) - c\triangle(n)) - J^c(\theta(n) + c\triangle(n))}{2c\triangle_i(n)}\right) + a(n)\xi(n)),$$

where $\xi(n) = o(1)$. Using Taylor series expansions of $J^c(\theta(n) - c\triangle(n))$ and $J^c(\theta(n) + c\triangle(n))$ around the point $\theta(n)$, one obtains

$$J^c(\theta(n) - c\triangle(n)) = J^c(\theta(n)) - c\sum_{j=1}^{N} \triangle_j(n)\nabla_j J^c(\theta(n)) + o(c)$$

and

$$J^c(\theta(n) + c\triangle(n)) = J^c(\theta(n)) + c\sum_{j=1}^{N} \triangle_j(n)\nabla_j J^c(\theta(n)) + o(c),$$

respectively. Thus,

$$\frac{J^c(\theta(n) - c\triangle(n)) - J^c(\theta(n) + c\triangle(n))}{2c\triangle_i(n)} = -\nabla_i J^c(\theta(n)) - \sum_{j=1, j\neq i}^{N} \frac{\triangle_j(n)}{\triangle_i(n)}\nabla_j J^c(\theta(n)) + o(c). \tag{12}$$

Now let $\{\mathcal{F}_n, n \geq 0\}$ denote the sequence of σ-fields defined by $\mathcal{F}_n = \sigma(\theta(m), m \leq n, \triangle(m), m < n)$, with $\triangle(-1) = 0$. It is now easy to see that the processes $\{M_1(n)\}, \ldots, \{M_N(n)\}$ defined by

$$M_i(n) = \sum_{m=0}^{n-1} a(m) \left(\frac{J^c(\theta(m) - c\triangle(m)) - J^c(\theta(m) + c\triangle(m))}{2c\triangle_i(m)} - E[\frac{J^c(\theta(m) - c\triangle(m)) - J^c(\theta(m) + c\triangle(m))}{2c\triangle_i(m)} \mid \mathcal{F}_m] \right),$$

$i = 1, \ldots, N$, form convergent martingale sequences. Thus, from (12), we have

$$E[\frac{J^c(\theta(n) - c\triangle(n)) - J^c(\theta(n) + c\triangle(n))}{2c\triangle_i(n)} \mid \mathcal{F}_n] = -\nabla_i J^c(\theta(n)) - \sum_{j=1, j\neq i}^{N} E[\frac{\triangle_j(n)}{\triangle_i(n)}]\nabla_j J^c(\theta(n)) + o(c).$$

By condition (A), $E[\frac{\triangle_j(n)}{\triangle_i(n)}] = 0$, $\forall j \neq i$, $n \geq 0$. The iteration in step 2 of the new algorithm can now be written as follows: for $i = 1, \ldots, N$,

$$\theta_i(n+1) = \Gamma_i^c(\theta_i(n) - a(n)\nabla_i J^c(\theta(n)) + a(n)\beta(n)), \tag{13}$$

where $\beta(n) = o(1)$ by the above. Thus, the iteration in step 2 can be seen as a discretization of the ODE (8), except for some additional error terms that, however, vanish asymptotically. Now for the ODE (8), K corresponds to the set of asymptotically stable fixed points, with $J^c(\theta)$ itself as an associated strict Liapunov function. The claim follows. □

Finally, note that we have $J(\theta) = J^c(\theta)$ for $\theta \in D$. Moreover, while updating θ in the original algorithm, we impose the restriction (via the projection $\Gamma(\cdot)$) that θ can move only on the grid D of points where $D = D^c \cap \mathcal{Z}^N$. By the foregoing analysis, the continuous parameter version of the algorithm shall converge in a "small" neighborhood of a local minimum of $J^c(\cdot)$. Thus, in the original algorithm, θ shall converge to a point on the grid that is (again) in a neighborhood of the local minimum (above) of $J^c(\cdot)$ and hence in a corresponding neighborhood of a local minimum of $J(\cdot)$; see Gerencsér, Hill, and Vágó [1, p. 1793] for a similar argument based on certain L-mixing processes. We now present our numerical experiments.

4. Numerical Experiments

We consider the problem of admission control in communication networks under bursty traffic. We specifically consider two different settings here. In both settings, we assume feedback policies to be of the threshold type that are, in particular, functions of the state of the underlying Markov process.

4.1 Setting 1

The basic model is shown in Figure 1. There is a single bottleneck node that is fed with arrivals that follow a Markov-modulated Poisson process (MMPP). In an MMPP, there is an underlying continuous time Markov chain (CTMC) $\{z_t\}$ taking values in a finite set (say) U. When $z_t = i$ (for some $i \in S$), the MMPP stream sends packets according to a Poisson process with rate $\lambda(i)$ that is a function of i. In general, we assume $\lambda(i) \neq \lambda(j)$ for $i \neq j$. The Markov chain $\{z_t\}$ stays for an exponential amount of time in a given state, at the end of which it transits to a new state. An MMPP is generally used to model bursty traffic in communication networks. The states in the underlying Markov chain may represent different classes of traffic. Note that if q_t denotes the queue length at time t, then $\{(q_t, z_t)\}$ corresponds to the underlying continuous time Markov chain for the whole system. We assume $\{z_t\}$ to have five states, numbered $0, 1, \ldots, 4$, with transition rate matrix Q given by

$$Q = \begin{bmatrix} -0.9 & 0.2 & 0.3 & 0.1 & 0.3 \\ 0.3 & -0.7 & 0.1 & 0.2 & 0.1 \\ 0.6 & 0.2 & -0.9 & 0 & 0.1 \\ 0.4 & 0.5 & 0.3 & -1.3 & 0.1 \\ 0.2 & 0.6 & 0.3 & 0.8 & -1.9 \end{bmatrix}.$$

The process $\{z_t\}$, for instance, spends an amount of time distributed according to exp(0.9) when in state 0 before transiting to some other state (see Q-matrix above). The

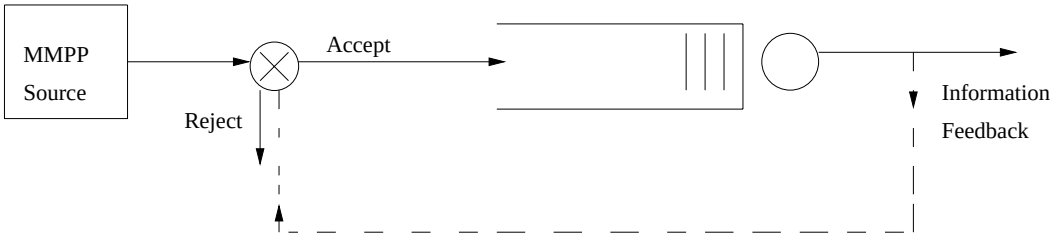

Figure 1. The admission control model of setting 1

rates of the Poisson process in each of the states 0 through 4 are taken to be 10, 15, 18, 22, and 30, respectively. The service times are assumed to be exponentially distributed with rate $\mu = 17$. In general, these rates could be set in any manner since we only require the Markov chain $\{(q_t, z_t)\}$ to be ergodic for any given set of threshold values (see below). Since we consider a finite buffer system, the above Markov chain will be ergodic. We consider the following class of feedback policies:

For $i = 0, 1, \ldots, 4$,
{
if MMPP state $= i$
{
if queue length $< N_i$
accept incoming packet;
else
reject incoming packet;
}
}

The parameter to be tuned is $(N_0, N_1, \ldots, N_4)^T$, where N_i corresponds to the threshold queue length when the state of the underlying CTMC is $i \in \{0, \ldots, 4\}$. The aim is to find the optimum parameter that would then give the corresponding optimal feedback policy within the class of policies described above. For this, we use our discrete parameter optimization algorithm described in section 2. Note that the form of feedback policies that we consider (see also setting 2 below) is quite general as it is based on the state of the underlying MMPP as well. In cases where such information is not available, one may consider optimizing a single threshold based on just the queue length at the bottleneck node. This would correspond to optimizing parameterized hidden Markov models or Markov chains under partial observations and could be easily handled using our algorithm [17].

The projection set D is chosen to be $\{2, \ldots, 490\}^5$. Thus, all the thresholds take values in the set $\{2, \ldots, 490\}$. The maximum number of customers in queue at any instant is thus upper bounded by 490 because of the form of the feedback policies and the constraint set. We assume that the queue length information is fed back to the controller once in every 100 arriving packets. The values of c and L in the algorithm are chosen to be 1 and 100, respectively. The step sizes $\{a(n)\}$ and $\{b(n)\}$ in the algorithm are taken as

$$b(n) = \frac{1}{\left[\frac{n}{10}\right]^{2/3}} \text{ and } a(n) = \frac{1}{\left[\frac{n}{10}\right]^{3/4}}, \qquad (14)$$

respectively, for $n \geq 1$ and $a(0) = b(0) = 1$. In the above, $\left[\frac{n}{10}\right]$ denotes the integer part of $\frac{n}{10}$. Note that we study the sensitivity of the algorithm w.r.t. the parameters c, L and the step sizes for setting 2 below. We consider $\triangle_i(n)$, $i = 0, 1, \ldots, 4$, $n \geq 0$ to be i.i.d., Bernoulli distributed with $\triangle_i(n) = \pm 1$ w.p. 1/2. The cost of accepting an incoming packet is the queue length as seen by it. For the rejection cost (RC), we assign a fixed constant value. We run the algorithm for 100,000 iterations or parameter updates. After convergence of the algorithm, the resulting threshold values are then used in another simulation that is run for 100,000 arrivals to compute the estimate for long-run average cost.

In Figure 2, we show the plot of convergence of threshold N_0 using our algorithm when the cost of rejection is set at 250. We do not show plots of other thresholds and also those corresponding to other rejection costs since these are somewhat repetitive in nature. In Table 1, we show values of threshold parameters to which our algorithm converged along with the long-run average costs obtained using the converged values for varying rejection costs but with cost of acceptance the same as above. Note also from Table 1 that on the whole, as the rejection cost is increased, the long-run average costs also increase. There is, however, no particular order in which the various thresholds converge. For the CTMC rate matrix given above, the steady-state probabilities π_i, $i = 0, 1, \ldots, 4$, are found to be $\pi_0 = 0.28$, $\pi_1 = 0.21$, $\pi_2 = 0.19$, $\pi_3 = 0.16$, and $\pi_4 = 0.16$, respectively. We also performed exper-

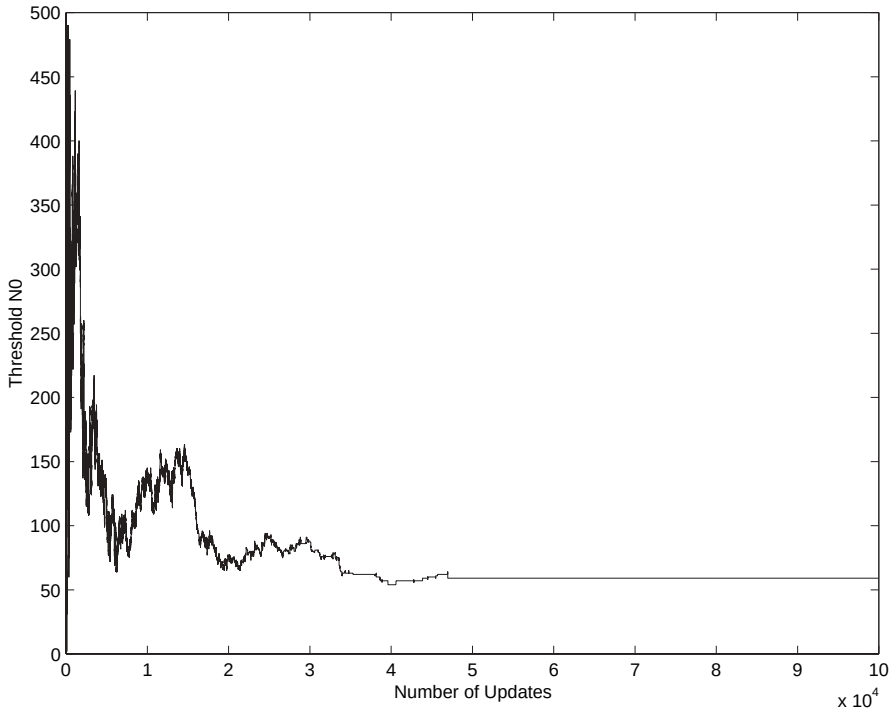

Figure 2. Convergence of threshold N_0 when rejection cost = 250

Table 1. Effect of increase in RC on performance

RC	N_0^*	N_1^*	N_2^*	N_3^*	N_4^*	$J(\theta^*)$
100	117	89	147	7	343	18.06
125	205	362	252	3	101	21.72
150	157	140	177	4	297	23.06
200	136	402	294	10	330	24.55
250	59	72	32	32	481	24.94
300	41	321	22	61	369	27.92
350	30	336	9	55	421	36.09
400	34	409	16	59	415	34.06
450	30	420	7	34	406	46.18
500	10	451	24	86	367	44.73

iments with different rejection costs in different states of the CTMC but with the cost of accepting arrivals the same as before. Specifically, we chose the rejection costs for this case to be $100 + 50i$ for states $i = 0, 1, \ldots, 4$ of the underlying CTMC. The average rejection cost in this case is thus 185.5. The long-run average cost for this case was found to be 23.80. Upon comparison with the values in Table 1, it is clear that this value lies in between the long-run average costs corresponding to the cases when the rejection costs are set uniformly (over all states) at 150 and 200, respectively, which conforms with intuition.

4.2 Setting 2

The model we consider here is in many ways more general than the one earlier. The basic model is shown in Figure 3. A similar model has recently been considered in Bhatnagar and Reddy [28], where the admission control problem is analyzed in detail and a variant of our algorithm that updates parameters in the (continuously valued) closed convex hull of the discrete search space is proposed. We consider two streams of packets, one controlled and the other uncontrolled. The uncontrolled

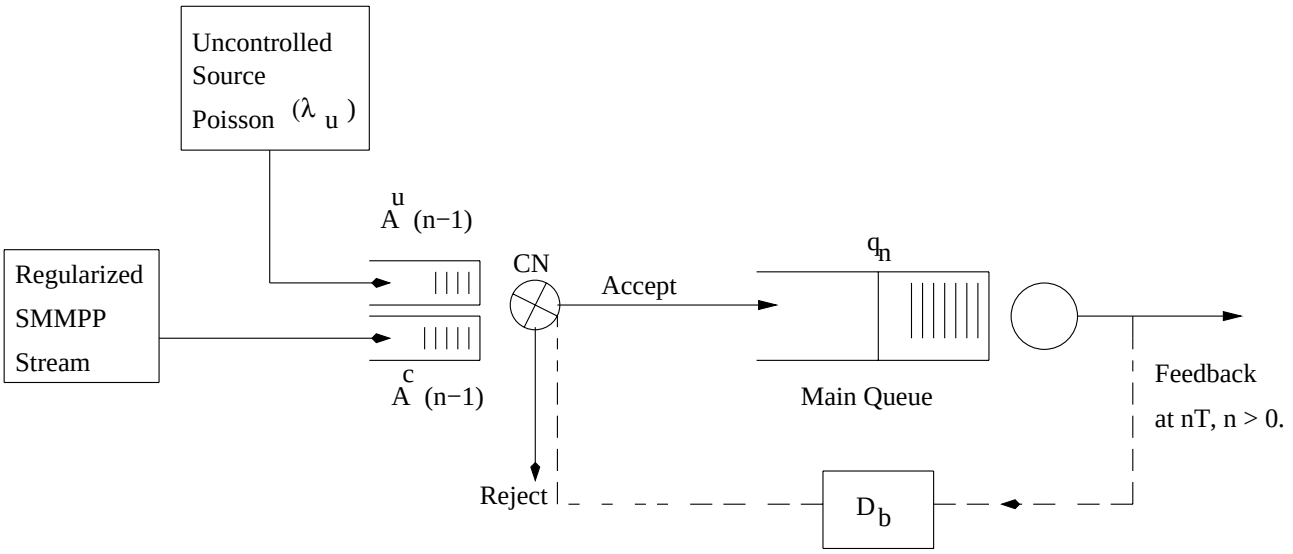

Figure 3. The admission control model of setting 2

stream corresponds to higher priority traffic, while the controlled stream characterizes lower priority traffic that is bursty in nature and receives best-effort service. We assume the uncontrolled stream to be a Poisson process with rate λ_u while the controlled stream is assumed to be a regularized semi-Markov-modulated Poisson process (SMMPP) that we explain below. Suppose $\{y_t\}$ is a regularized semi-Markov process, that is, one for which transitions are Markov but that take place every T instants of time for some T fixed. Suppose $\{y_t\}$ takes values in a finite set V. Now if $y_n \equiv y_{nT} = i \in V$, we assume the SMMPP sends packets according to a Poisson process with rate $\lambda(i)$ during the time interval $[nT, (n+1)T)$. As with an MMPP, an SMMPP may also be used to model bursty traffic.

Packets from both streams are first collected at a multiplexor or control node CN during time intervals $[(n-1)T, nT), n \geq 1$, and are stored in different buffers. (These correspond to the input buffers at node CN.) We assume both buffers have infinite capacity. At instants nT, the queue length $q_n, n \geq 1$ in the main queue is observed, and this information is fed back to node CN. We assume, however, that this information is received at the CN with a delay D_b. On the basis of this information, a decision on the number of packets to accept from both streams (that are stored in the control buffers) is instantly made. Packets that are not admitted to the main queue from either stream are immediately dropped, and they leave the system. Thus, the interim buffers at the control node CN are emptied every T units of time, with packets from these buffers either joining the main queue or getting dropped. We assume that packets from the uncontrolled source waiting at node CN are admitted to the main queue first (as they have higher priority) whenever there are vacant slots available in the main queue buffer that we assume has size B. Admission control (in the regular sense) is performed only on packets from the SMMPP stream that are stored at node CN. Packets from this stream are admitted to the main queue at instants nT based on the queue length q_n of the main queue, the number of uncontrolled stream packets admitted at time nT, and also the state y_{n-1} (during interval $[(n-1)T, nT)$) of the underlying Markov process in the SMMPP. We assume that feedback policies (for the controlled stream) are of the threshold type. For D_b of the form $D_b = MT$ for some $M > 0$, the joint process $\{(q_n, y_{n-1}, q_{n-1}, y_{n-2}, \ldots, q_{n-M}, y_{n-M-1})\}$ is Markov. For ease of exposition, we give the form of the policies below for the case of $D_b = 0$. Suppose $\bar{L}_1, \ldots, \bar{L}_N$ are given integers satisfying $0 \leq \bar{L}_j \leq B, \forall\ j \in V$. Here, each $\bar{L}_j$ serves as a threshold for the controlled SMMPP stream when the state of the underlying Markov chain is j. Let a^u and a^c denote the number of arrivals from the uncontrolled and controlled streams, respectively, at node CN during the time interval $[(n-1)T, nT)$.

Feedback Policies

- If $a^u \geq B - i$
  {
  Accept first $(B - i)$ uncontrolled packets and no controlled packets.
  }
- If $i < \bar{L}_j$ and $\bar{L}_j - i \leq a^u < B - i$
  {

Accept all uncontrolled packets and no controlled packets.
}

- If $i < \bar{L}_j$ and $\bar{L}_j - i > a^u$
  {
  Accept all uncontrolled packets and
  - If $a^c < \bar{L}_j - i - a^u$
    {
    Accept all controlled packets
    }
  - If $a^c \geq \bar{L}_j - i - a^u$
    {
    Accept first $\bar{L}_j - i - a^u$ controlled packets
    }

  }
- If $i \geq \bar{L}_j$ and $B - i > a^u$
  {
  Accept all uncontrolled packets and no controlled packets.
  }

For our experiments, we consider an SMMPP stream with four states, with the transition probability matrix of the underlying Markov chain chosen to be

$$P = \begin{bmatrix} 0 & 0.6 & 0.4 & 0 \\ 0.4 & 0 & 0.6 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0.7 & 0 & 0.3 \end{bmatrix}.$$

The corresponding rates of the associated Poisson processes in each of these states are chosen as $\lambda(1) = 0.5$, $\lambda(2) = 1.0$, $\lambda(3) = 1.5$, and $\lambda(4) = 2.5$, respectively. The service times in the main queue are considered to be exponentially distributed with rate $\mu = 1.3$. We show results of experiments with different values of T, λ_u, D_b, and the rejection cost (RC). These quantities are assumed constant for a given experiment. We also show experiments with different values for step sizes $\{a(n)\}$, and $\{b(n)\}$, as well as parameters c and L, respectively.

In Figure 4, we show the convergence behavior for one of the threshold values, for the case $T = 3$, $\lambda_u = 0.4$, $D_b = 0$, and $RC = 50$. The same for other cases and thresholds is similar and is hence not shown. We denote by $N1^*, \ldots, N4^*$ the optimal values of the threshold parameters $N1, \ldots, N4$, respectively. We assume that the cost of accepting a packet at the main queue is the queue length as seen by it. The buffer size at the main queue is assumed to be 80.

In Tables 2 and 3, we study the change in average cost behavior for different values of T and λ_u when $RC = 50$ and $D_b = 0$. Specifically, in Table 2 (respectively, Table 3), the results of experiments for which $\lambda_u = 0.1$ (respectively, 1.0) are tabulated. Note that for a given value of λ_u, the average cost increases as T is increased (and thus is the lowest for the lowest value of T considered). This happens because as T is increased, the system exercises control less often as also the average number of packets that arrive into the control node buffers in any given interval increases. Also from Table 3, note that as the uncontrolled rate λ_u is increased to 1.0, the average cost values increase significantly over corresponding values of these for $\lambda_u = 0.1$. This is again expected since the higher priority uncontrolled packets now have a significant presence in the main queue buffer, leading to an overall increase in values of $J(\theta^*)$ as packets from the control stream get rejected more frequently. In Table 4, we vary RC over a range of values, keeping the cost of accepting a packet the same as before. Also, we keep T and λ_u fixed at 3 and 0.4, respectively. Furthermore, $D_b = 0$. As expected, the average cost $J(\theta^*)$ increases as RC is increased. In Table 5, we show the impact of varying D_b for fixed $T = 3$, $\lambda_u = 0.1$, and $RC = 50$, respectively. Note that as D_b is increased, performance deteriorates as expected.

In the above tables, we set the step sizes as in (14). Also, $c = 1$ and $L = 100$ as before. In Tables 6 through 8, we study the effect of changing these parameters when $T = 3$ and $RC = 50$ are held fixed. In Table 6, we let $b(n)$ as in (14), while $a(n)$ has the form $a(0) = 1$ and $a(n) = \frac{1}{\left[\frac{n}{10}\right]^{\alpha}}$, $n \geq 1$. We study the variation in performance for different values of $\alpha \in [0.66, 1]$. Note from the table that when $\alpha = 0.66$ is used, $a(n) = b(n)$, $\forall n$. In such a scenario, the algorithm does not show good performance as observed from the average cost value, which is high. When α is increased, performance improves very quickly until after $\alpha = 0.75$, beyond which it stabilizes. These results also suggest that a difference in timescales in the recursions improves performance.

In Tables 7 and 8, we study the effects of changing c and L on performance for above values of T and RC, with step sizes as in (14). Note from Table 7 that performance is not good for $c < 0.5$. As described previously (section 2), this happens because for $c < 0.5$, the parameters for both simulations after projection would lead to the same point. Performance is seen to improve for $c > 0.5$. From Table 8, it can be seen that performance is not good for low values of L (see the case corresponding to $L = 10$). It has also been observed in the case of continuous optimization algorithms [17, 19, 20] that low values of L lead to poor system performance. Also, observe that when L is increased beyond a point (see entries corresponding to $L = 700$ and $L = 1000$, respectively), performance degrades again. This implies that excessive additional averaging is also not good for system performance. From the table, it can be seen that L should ideally lie between 100 and 500.

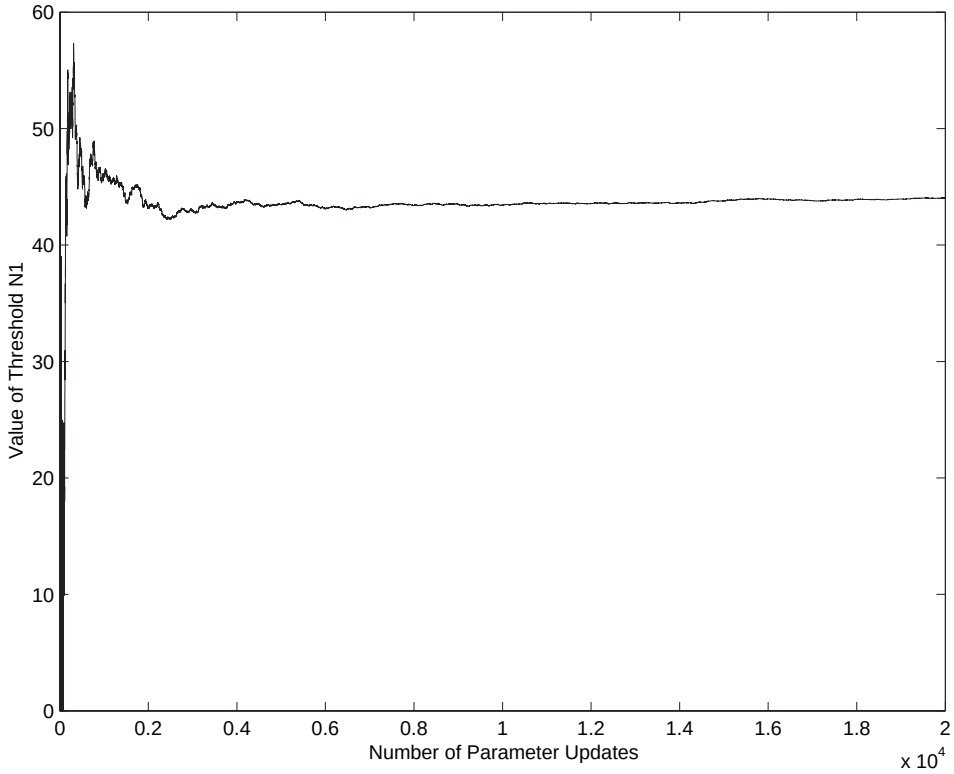

Figure 4. Convergence of the values of threshold $N1$ for $T = 3$, $\lambda_u = 0.4$, $D_b = 0$, and $RC = 50$

Table 2. $\lambda_u = 0.1$, $RC = 50$, $D_b = 0$

T	$N1^*$	$N2^*$	$N3^*$	$N4^*$	$J(\theta^*)$
3	15	23	36	4	14.23
5	11	9	38	21	15.92
10	23	20	11	59	18.08

Table 3. $\lambda_u = 1.0$, $RC = 50$, $D_b = 0$

T	$N1^*$	$N2^*$	$N3^*$	$N4^*$	$J(\theta^*)$
3	8	41	28	19	25.69
5	14	11	19	6	27.91
10	21	19	37	9	31.32

Table 4. $T = 3$, $\lambda_u = 0.4$, $D_b = 0$

RC	$N1^*$	$N2^*$	$N3^*$	$N4^*$	$J(\theta^*)$
40	23	16	36	17	15.12
50	44	32	23	19	16.81
60	26	36	14	30	19.40
70	31	25	19	12	22.08
80	22	29	49	53	25.59
90	19	43	42	36	29.61
100	8	50	38	26	34.18

Table 5. $\lambda_u = 0.4$, $T = 3$, $RC = 50$

D_b	$N1^*$	$N2^*$	$N3^*$	$N4^*$	$J(\theta^*)$
0	44	32	23	19	16.81
3	53	35	41	9	18.33
6	11	48	50	10	19.52
9	12	56	17	12	21.64

Table 6. $T = 3$, $\lambda_u = 0.1$, $RC = 50$, $D_b = 0$

α	$N1^*$	$N2^*$	$N3^*$	$N4^*$	$J(\theta^*)$
0.66	59	26	6	19	25.12
0.70	37	32	19	21	18.81
0.72	21	31	26	28	16.42
0.75	15	23	36	4	14.23
0.80	24	32	37	12	13.67
0.85	16	29	49	43	13.79
0.90	14	37	32	29	13.61
1.00	18	45	38	16	13.66

Table 7. $T = 3$, $\lambda_u = 0.1$, $RC = 50$, $D_b = 0$

c	$N1^*$	$N2^*$	$N3^*$	$N4^*$	$J(\theta^*)$
0.40	58	50	32	40	29.12
0.50	52	38	13	19	18.29
0.52	19	20	38	10	14.04
0.55	11	19	29	21	13.71
0.70	21	28	35	17	14.42
1.00	15	23	36	4	14.23

Table 8. $T = 3$, $\lambda_u = 0.1$, $RC = 50$, $D_b = 0$

L	$N1^*$	$N2^*$	$N3^*$	$N4^*$	$J(\theta^*)$
10	51	39	21	45	29.20
50	32	28	22	29	15.29
100	15	23	36	4	14.23
200	9	15	29	8	13.92
500	17	18	25	47	14.01
700	27	9	34	31	16.22
1000	21	38	4	51	18.26

5. Conclusions

We developed in this article a two-timescale simultaneous perturbation stochastic approximation algorithm that finds the optimum parameter in a discrete search space. This algorithm is applicable in cases where the cost to be optimized is in itself the long-run average of certain cost functions whose noisy estimates can be obtained using simulation. The advantage of a simulation-based approach is that the transition dynamics of the underlying Markov processes need not be precisely known as long as state transitions of these can be simulated. We briefly presented the convergence analysis of our algorithm. Finally, we applied this algorithm to find optimal feedback policies within threshold-type policies for an admission control problem in communication networks. We showed experiments under two different settings. The results obtained are along expected lines. The proposed algorithm, however, must be tried on high-dimensional parameter settings and tested for their online adaptability in large simulation scenarios. In general, SPSA algorithms are known to scale well under such settings; see, for instance, Bhatnagar et al. [17] and Bhatnagar [20], who consider high-dimensional parameters. The same, however, should be tried for the discrete parameter optimization algorithm considered here.

Recently, in Bhatnagar et al. [19], SPSA-type algorithms that use certain deterministic perturbations instead of randomized have been developed. These are found to improve performance over randomized perturbation SPSA. Along similar lines, in Bhatnagar and Borkar [18], the use of a chaotic iterative sequence for random number generation for averaging in the generated perturbations in SPSA algorithms has been proposed along with a smoothed functional algorithm that uses certain perturbations involving Gaussian random variables. In Bhatnagar [20], adaptive algorithms for simulation optimization based on the Newton algorithm have also been developed for continuous parameter optimization. It would be interesting to develop discrete parameter analogs of these algorithms as well and to study their performance in other applications.

6. Acknowledgments

The authors thank all the reviewers for their many comments that helped in significantly improving the quality of this manuscript. The authors thank Mr. I. B. B. Reddy for help with the simulations. The first author was supported in part by grant no. SR/S3/EE/43/2002-SERC-Engg from the Department of Science and Technology, Government of India.

7. References

[1] Gerencsér, L., S. D. Hill, and Z. Vágó. 1999. Optimization over discrete sets via SPSA. In *Proceedings of the IEEE Conference on Decision and Control*, pp. 1791-5.

[2] Gokbayrak, K., and C. G. Cassandras. 2001. Online surrogate problem methodology for stochastic discrete resource allocation problems. *Journal of Optimization Theory and Applications* 108 (2): 349-75.

[3] Barnhart, C. M., J. E. Wieselthier, and A. Ephremides. 1995. Admission-control policies for multihop wireless networks. *Wireless Networks* 1:373-87.

[4] Kirkpatrick, S., C. D. Gelatt, and M. Vecchi. 1983. Optimization by simulated annealing. *Science* 220:621-80.

[5] Gelfand, S. B., and S. K. Mitter. 1989. Simulated annealing with noisy or imprecise energy measurements. *Journal of Optimization Theory and Applications* 62 (1): 49-62.

[6] Alrefaei, M. H., and S. Andradóttir. 1999. A simulated annealing algorithm with constant temperature for discrete stochastic optimization. *Management Science* 45 (5): 748-64.

[7] Yan, D., and H. Mukai. 1992. Stochastic discrete optimization. *SIAM Journal of Control and Optimization* 30 (3): 594-612.

[8] Gong, W.-B., Y.-C. Ho, and W. Zhai. 1999. Stochastic comparison algorithm for discrete optimization with estimation. *SIAM Journal of Optimization* 10 (2): 384-404.

[9] Norkin, V. I., Y. M. Ermoliev, and A. Ruszczyński. 1998. On optimal allocation of indivisibles under uncertainty. *Operations Research* 46 (3): 381-95.

[10] Jordan, S., and P. Varaiya. 1994. Control of multiple service multiple resource communication networks. *IEEE Transactions on Communications* 42 (11): 2979-88.

[11] Cassandras, C. G., and C. G. Panayiotou. 1999. Concurrent sample path analysis of discrete event systems. *Journal of Discrete Event Dynamic Systems: Theory and Applications* 9:171-95.

[12] Ho, Y.-C., and X.-R. Cao. 1991. *Perturbation analysis of discrete event dynamical systems*. Boston: Kluwer.

[13] Gokbayrak, K., and C. G. Cassandras. 2002. Generalized surrogate problem methodology for online stochastic discrete optimization. *Journal of Optimization Theory and Applications* 114 (1): 97-132.

[14] L'Ecuyer, P., and P. W. Glynn. 1994. Stochastic optimization by simulation: Convergence proofs for the $GI/G/1$ queue in steady-state. *Management Science* 40 (11): 1562-78.

[15] Spall, J. C. 1992. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE Transactions on Automatic Control* 37 (3): 332-41.

[16] Bhatnagar, S., and V. S. Borkar. 1998. A two time scale stochastic approximation scheme for simulation based parametric optimization. *Probability in the Engineering and Informational Sciences* 12:519-31.

[17] Bhatnagar, S., M. C. Fu, S. I. Marcus, and S. Bhatnagar. 2001. Two timescale algorithms for simulation optimization of hidden Markov models. *IIE Transactions* 33 (3): 245-58.

[18] Bhatnagar, S., and V. S. Borkar. 2003. Multiscale chaotic SPSA and smoothed functional algorithms for simulation optimization. *SIMULATION* 79 (10): 568-80.

[19] Bhatnagar, S., M. C. Fu, S. I. Marcus, and I.-J. Wang. 2003. Two-timescale simultaneous perturbation stochastic approximation using deterministic perturbation sequences. *ACM Transactions on Modelling and Computer Simulation* 13 (2): 180-209.

[20] Bhatnagar, S. 2005. Adaptive multivariate three-timescale stochastic approximation algorithms for simulation based optimization. *ACM Transactions on Modeling and Computer Simulation* 15 (1): 74-107.

[21] Kelly, F. P., P. B. Key, and S. Zachary. 2000. Distributed admission control. *IEEE Journal on Selected Areas in Communications* 18:2617-28.
[22] Liebeherr, J., D. E. Wrege, and D. Ferrari. 1996. Exact admission control for networks with a bounded delay service. *IEEE/ACM Transactions on Networking* 4 (6): 885-901.
[23] Bertsekas, D. P., and J. N. Tsitsiklis. 1996. *Neuro-dynamic programming*. Belmont, MA: Athena Scientific.
[24] Marbach, P., and J. N. Tsitsiklis. 2001. Simulation-based optimization of Markov reward processes. *IEEE Transactions on Automatic Control* 46 (2): 191-209.
[25] Schweitzer, P. J. 1968. Perturbation theory and finite Markov chains. *Journal of Applied Probability* 5:401-13.
[26] Vazquez-Abad, F. J., and H. J. Kushner. 1992. Estimation of the derivative of a stationary measure with respect to a control parameter. *Journal of Applied Probability* 29:343-52.
[27] Hirsch, M. W. 1989. Convergent activation dynamics in continuous time networks. *Neural Networks* 2:331-49.
[28] Bhatnagar, S., and I. B. B. Reddy. 2005. Optimal threshold policies for admission control in communication networks via discrete parameter stochastic approximation. *Telecommunication Systems* 29 (1): 9-31.

Shalabh Bhatnagar *is an assistant professor in the Department of Computer Science and Automation at the Indian Institute of Science, Bangalore, India.*

Hemant J. Kowshik *is an undergraduate student in the Department of Electrical Engineering at the Indian Institute of Technology Madras, Chennai, India.*