Proceedings of the 1990 Winter Simulation Conference
Osman Balci, Randall P. Sadowski, Richard E. Nance (eds.)

FINITE-TIME BEHAVIOR OF TWO SIMULATION OPTIMIZATION ALGORITHMS

Ying Tat Leung

Philips Laboratories
North American Philips Corporation
345 Scarborough Road
Briarcliff Manor, New York 10510

Rajan Suri

Department of Industrial Engineering
University of Wisconsin - Madison
1513 University Avenue
Madison, Wisconsin 53706

ABSTRACT

We investigate the finite-time behavior of two specific simulation optimization algorithms: a Robbins-Monro procedure applied in a conventional way and a more recently proposed single-run optimization algorithm. By applying these algorithms to simple systems we show that, in practice, convergence of the former algorithm can be slow while that of the latter is very fast. We also provide evidence that the choice of projection operator (to deal with constraints in the optimization problem) has a significant effect on the finite-time performance of the latter algorithm. These results provide some basic insight into the behavior of such algorithms.

1. INTRODUCTION

We consider optimizing an objective function which involves quantities that can only be observed through running a computer simulation model (or by observing the real world). This situation often arises when one is interested in optimizing the performance of a complex system such as a manufacturing facility. We assume that random components exist in the system, so that the computer simulation model only gives sample estimates of the quantities of interest. Our optimization problem therefore falls into the category of stochastic optimization (e.g. [Glynn 1986]). Pre-1977 works on stochastic optimization via simulation are surveyed extensively in [Farrell 1977], while more recent survey papers are those of Meketon [1987] and Jacobson and Schruben [1989].

An important class of stochastic optimization algorithms is stochastic approximation, first proposed by Robbins and Monro [1951]. Their original algorithm is not an optimization scheme, but rather a root finding procedure for a response function whose exact values are not known but are observed with noise. In an optimization scenario, the response function represents the gradient of an objective function. If we can find an unbiased estimator for the response function, it is well-known that the asymptotic convergence rate for the Robbins-Monro algorithm is most often $T^{-1/2}$, where T is the computational effort. This is usually the best we can expect from Monte-Carlo algorithms.

However, asymptotic convergence rates alone may not give sufficient information on how fast the algorithms actually converge in practice. This is because in general we do not know when the algorithms will start to behave "asymptotically". Therefore two algorithms with the same asymptotic behavior may perform quite differently when they are actually implemented and run for a finite period of time. Recent empirical studies on a class of simulation optimization algorithms, namely single-run optimization [Suri and Zazanis 1988, L'Ecuyer et al. 1989, Suri and Leung 1989], illustrate this point well. Results there suggest that single-run optimization algorithms converge much faster than other conventional simulation optimization algorithms, even though we know that the asymptotic convergence rate cannot be faster than $T^{-1/2}$. From a practical point of view, the finite-time behavior of an algorithm is at least as important as the asymptotic behavior. Not only does the finite-time behavior give a sense of how far the current solution is from the true optimum, but in so doing it also suggests when to stop execution of the algorithm. Needless to say, such information is very useful for practitioners.

Analyzing the finite-time performance of stochastic optimization algorithms is in general a very hard problem. (In fact, establishing asymptotic properties is by no means easy, see e.g. [Ljung 1977, Kushner and Clark 1978].) In this paper we use some simple models to investigate the finite-time behavior of two stochastic approximation procedures adapted to the simulation environment. These models are believed to be representative of many simulation models and yet simple enough to give insightful results on the finite-time behavior of the optimization algorithms. In addition, using one of the models we study the impact of the projection operator (to deal with constraints) on the performance of the algorithm. Since a projection operator is most often carried out a finite number of times (in the case of convergent algorithms), it does not affect the asymptotic convergence rate of the algorithm. However, we show that a projection operator does contribute to the finite-time behavior and as a result is an important consideration in practice.

Throughout the paper we let x be a real-valued controllable or decision parameter of a simulation model, x^* be the value of x we are seeking (e.g. root of the gradient of the objective function), and $x(n)$ be its estimate generated by the optimization procedure in the nth iteration.

2. STOCHASTIC APPROXIMATION FOR A LINEAR RESPONSE MODEL

2.1 The Linear Response Model

In this section we consider conventional stochastic approximation for root finding, namely the Robbins-Monro procedure [Robbins and Monro 1951]. It is well known that Robbins-Monro type procedures when combined with suitable estimators typically have an asymptotic convergence rate of $T^{-1/2}$, where T is the computational effort. This has been shown to be true in a number of papers under a variety of conditions (e.g. [Sacks 1958, Major and Révész 1973, Kushner and Clark 1978, Kushner and Huang 1979]).

On the other hand, little has been done on the finite-time behavior of stochastic approximation. When the estimator for the underlying response function is unbiased and under certain additional conditions, Chung [1954] gives a bound on the second moment of $x(n)$ centered around x^*. Here we relax the assumption of an unbiased estimator but only consider cases where the underlying response function is linear. By obtaining an explicit expression for the mean squared error for the estimate of x^* at the end of the nth iteration, we can gain some insight into the behavior of the algorithm when the number of iterations is small.

Let $H(x)$ denote a real-valued function representing the underlying response function of a simulation model with a real-valued controllable parameter x. Suppose $H(x)$ is unknown to the experimenter who can only observe noisy values of it at different values of x. The goal of the experimenter is to solve for x^* such that $H(x^*) = \alpha$, where α is a given constant. Assume that the experimenter chooses to use the following Robbins-Monro type procedure.

Algorithm 1. A Robbins-Monro procedure

(1) Initialize: Choose $x(1) = \zeta$, ζ arbitrary, a fixed initial state s_0 of the system to be simulated, a sequence of positive numbers $\{a(k), k = 1, 2, \ldots\}$ such that condition (A2.1.3) below is satisfied, and another sequence of positive numbers $\{T(k), k = 1, 2, \ldots\}$ such that $T(k) \to \infty$ as $k \to \infty$. $T(k)$ is the simulation run length for iteration k. Set $n = 1$.
(2) Make an independent simulation run at $x(n)$ with initial state

s_0 and run length $T(n)$. Let $Y(n)$ be the (noisy) system response observed.

(3) Update:

$$x(n+1) = x(n) - a(n)[Y(n) - \alpha], \quad n = 1, 2, \ldots \qquad (1)$$

$$n = n + 1.$$

(4) If the chosen stopping criterion is not satisfied, go to step 2. Otherwise, stop the algorithm and $x(n)$ is an estimate of x^*.

[End of algorithm]

Algorithm 1 is a natural adaptation of the original Robbins-Monro procedure to the simulation environment and is similar to the one proposed by Wardi [1988]. We assume that a suitable stopping criterion (in step 4) has been chosen so that the algorithm eventually stops (see e.g. [Stroup and Braun 1982]).

We consider the case where $H(x)$ is of a linear form,

$$H(x) = \mu x - \theta, \qquad (2)$$

where μ and θ are constants and $\mu \neq 0$. Without loss of generality, we assume that $\mu > 0$ and $\alpha = 0$. The true solution is therefore $x^* = \theta/\mu$. Suppose that when given $x(1), \ldots, x(n)$, $Y(1), \ldots, Y(n-1)$; $Y(n)$ can be written as

$$Y(n) = H(x(n)) + \beta(n) + \xi(n), \qquad (3)$$

where $\beta(n)$ can be viewed as the bias due to the initial state and $\xi(n)$ the noise. In order to analyze this system, we further assume that

(A2.1.1) $\{\beta(n), n = 1, 2, \ldots\}$ is a sequence of bounded deterministic functions such that $\beta(n) \to 0$ as $n \to \infty$.

(A2.1.2) $\{\xi(n), n = 1, 2, \ldots\}$ is a sequence of independent and identically distributed (i.i.d.) random variables with zero mean and finite variance σ^2.

The additive form of (3) is also used in, for example, [Kushner and Clark 1978] and seems to be fairly general. Assumption (A2.1.1) says that the initial bias has to go to zero as the simulation run length $T(n)$ increases. Since we start each iteration with a fixed initial system state s_0, it is not unreasonable to assume $\beta(n)$ is deterministic (but may depend on s_0). A potential weakness of this assumption, however, is that we do not allow $\beta(n)$ to depend on the parameter value $x(n)$ at which the simulation is running. $\{\xi(n)\}$ stands for the random noise observed in independent simulation runs. For convergence of Algorithm 1, we also assume that

(A2.1.3) $\{a(n), n = 1, 2, \ldots\}$ is a deterministic sequence of positive real numbers such that $a(n) \to 0$ as $n \to \infty$, $\sum_n a(n) = \infty$, and $\sum_n [a(n)]^2 < \infty$.

In the optimization situation, $H(x)$ is the derivative of an objective function, and so (2) implies that the original objective function is quadratic. This is not unreasonable for many practical systems operating near a local optimum.

2.2 Finite-Time Behavior of the Algorithm

We are interested in the behavior of Algorithm 1 when the number of iterations n is finite. Assuming the system response has the form of (3) and that (A2.1.1) and (A2.1.2) hold, it can be shown [Leung 1990] that, for $n = 1, 2, \ldots$,

$$\begin{aligned} \mathrm{E}[x(n+1)] &= \frac{\theta}{\mu} + \left(\zeta - \frac{\theta}{\mu}\right) \prod_{k=1}^{n} [1 - \mu a(k)] - \\ &\quad \sum_{j=1}^{n} \left[\beta(j) a(j) \prod_{k=j+1}^{n} (1 - \mu a(k)) \right], \qquad (4) \\ \mathrm{var}[x(n+1)] &= \sigma^2 \sum_{k=1}^{n} \left[a^2(k) \prod_{j=k+1}^{n} (1 - \mu a(j))^2 \right]. \qquad (5) \end{aligned}$$

On the RHS of (4), note that $\theta/\mu = x^*$, so the second term is the bias due to the starting value ζ, and the last term is the bias due to that contained in the observations $Y(n)$ (namely $\beta(n)$). It is also intuitively clear that (5) is independent of $\beta(n)$ since $\beta(n)$ is assumed to be deterministic. From these two formulae, the mean squared error (MSE) of $x(n+1)$ is immediate from the relation

$$\begin{aligned} \mathrm{MSE}[x(n+1)] &= \mathrm{E}\left\{[x(n+1) - x^*]^2\right\} \\ &= \{\mathrm{E}[x(n+1)] - x^*\}^2 + \mathrm{var}[x(n+1)]. \end{aligned}$$

We give a numerical example of calculating the MSE. Let A denote a constant. We take $a(n)$ to be A/n, a common choice for Robbins-Monro type procedures, and $\beta(n)$ to be $1/n$. (It is well known that many practical simulation models have an initial bias inversely proportional to the simulation run length.) Other parameters used are: $\theta = 4$, $\mu = 2$ (hence $x^* = 2$), $\sigma^2 = 8$, $\zeta = 0$.

Figure 1 shows the graphs of MSE versus n for different values of A. It is clear that when $A = 0.125$, the MSE decreases much more slowly than that of the other cases. For a large value of A such as 2.125, the MSE in the first few iterations is very big but decreases quickly after 6 or 7 iterations. Other cases with different parameter values also show similar behavior and are omitted here.

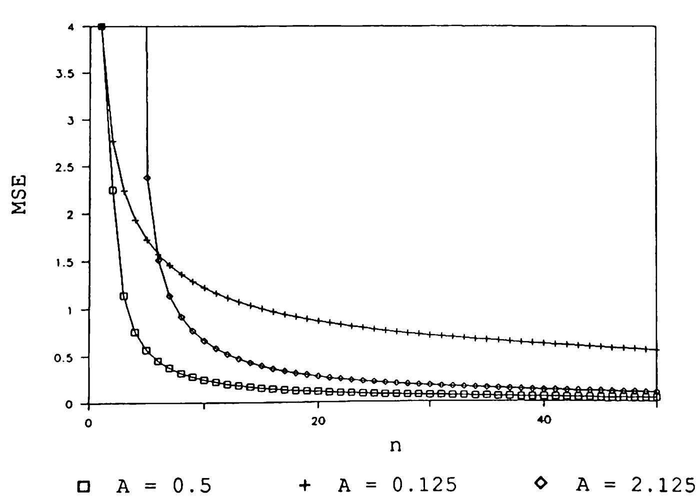

Figure 1. Behavior of Algorithm 1

An important lesson learned from this analysis is that although the Robbins-Monro procedure typically enjoys a fast (in terms of Monte Carlo algorithms) asymptotic convergence rate of $n^{-1/2}$, its finite-time behavior is not as good as we wish it to be. For example, if we want the estimate of x^* to be, on the average, within plus or minus 5% of the true value (i.e. $x(n) \in [1.9, 2.1]$, resulting in an MSE$[x(n)]$ of 0.01 and $H(x(n)) \in [-0.2, 0.2]$, reasonable requirements), the number of iterations needed is not small, even when using the better choices of $A = 0.5$ or 2.125. The exact numbers of iterations are shown in Table 1. Since each iteration involves one simulation run, the total computational effort is rather substantial, especially for models of complex systems.

Table 1. Small-Sample Behavior of Algorithm 1

A	σ^2	Smallest n such that MSE$[x(n)] \leq 0.01$
0.5	1	38
	8	206
2.125	1	64
	8	485

It should be noted that we have no intention of diminishing the value of the Robbins-Monro algorithm here. On the contrary, we feel that it is one of the best algorithms available for (continuous parameter) stochastic optimization. It is simple, usually does not have any numerical instability problems, and when applied carefully can be much faster than, say, techniques adapted from deterministic optimization methods. It also forms the basis of single-run optimization, as will be seen next.

3. SINGLE-RUN OPTIMIZATION FOR AN AR(1) PROBLEM

3.1 The Optimization Problem

In this section we apply the Robbins-Monro procedure within a simulation run, resulting in a single-run optimization algorithm. The optimization problem is in fact analytically tractable, but we shall use simulation to solve the problem as if this were not so in order to analyze the simulation optimization algorithm. Though simple, this problem illustrates some interesting results which seem counter-intuitive on first sight. For instance, the observations (from simulation) $Y(x(n))$ used in the optimization algorithm are biased and highly correlated; additionally, the value of $x(n)$ is updated after only one noisy observation is made. Yet the algorithm is convergent [Leung 1990] and possesses highly desirable small-sample behavior, as we shall show later.

It turns out that the optimization problem studied here reduces to a level crossing problem used by Meketon [1983, 1987] to demonstrate the single-run optimization concept. All notations used in this section are independent of those in section 2.

Consider an autoregressive process of order one (AR(1) process) which depends on a real-valued parameter x:

$$\begin{aligned} Y(0,x) &= 0, \\ Y(n,x) &= x \cdot Y(n-1,x) + \epsilon(n), \quad n = 1, 2, \ldots \end{aligned} \tag{6}$$

Let $\Gamma = \{x \mid -1 < x < 1\}$. We assume the following.
(A3.1.1) $x \in \Gamma$, and
(A3.1.2) $\{\epsilon(n), n = 1, 2, \ldots\}$ is a sequence of i.i.d. random variables having a Normal distribution with finite mean μ and variance σ^2.

We take this AR(1) process as a model of the output from a simulation run of a discrete-event stochastic system. For example, if we are simulating a single server queue and are interested in the waiting time per customer, then $Y(n,x)$ represents the waiting time of the nth customer and is assumed to obey (6). While the use of such a model necessarily involves considerable simplification, it is generally believed that the AR(1) process does possess many important statistical properties of the output processes commonly found in discrete-event simulations. For example, both $\mathrm{cov}(Y(n,x), Y(n+k,x))$ and the effect of $Y(0,x)$ on the conditional mean $\mathrm{E}[Y(k,x) \mid Y(0,x)]$ decreases exponentially to zero as $k \to \infty$. The class of AR(1) processes have long been used as a model for output processes from discrete-event simulations (e.g. [Fishman 1972, Turnquist and Sussman 1977, Kelton and Law 1984, Kelton 1986]).

Suppose the steady-state performance measure of interest is

$$M(x) = \lim_{n\to\infty} \mathrm{E}[Y(n,x)].$$

Let α be a constant. Our objective is to solve:

$$\min_{x\in\Gamma} \left(J(x, M(x)) = [M(x) - \alpha]^2 \right). \tag{7}$$

Assume, for the time being, that the minimum value of zero is achievable in the constraint set Γ. Then (7) is equivalent to solving

$$M(x) - \alpha = 0, \tag{8}$$

which is identical to that considered in [Meketon 1987]. Let x^* be a solution to (8). From standard results in the theory of time series, we know $M(x) = \mu/(1-x)$. So for $M(x^*) = \alpha$, we have

$$\alpha x^* - (\alpha - \mu) = 0.$$

Note that this is a linear equation in x^*, which is consistent with the assumption of a linear response function in section 2. We assume
(A3.1.3) $\alpha \neq 0$ and $-1 < (\alpha - \mu)/\alpha < 1$ so that an achievable minimum within Γ is ensured. Further, without loss of generality, we restrict α to be strictly positive. (The case of $\alpha < 0$ is analogous.)

3.2 The Single-Run Optimization Algorithm

If we were to use a conventional search procedure to solve this problem (e.g. Hooke and Jeeves' [1961] pattern search), we would first have to simulate a number of $Y(i,x)$'s, delete the transient observations, estimate $M(x)$ from the steady-state observations, and update the value of x based on the estimated value of $M(x)$. The procedure would then be repeated a number of times until an estimate of the optimum was obtained. Obviously, the total computational effort involved would be large.

Let u be a fixed positive real number close to but strictly less than 1 such that $x^* \in [-u, u]$, unif$[a,b]$ be an independent uniform random variate between a and b, and K be a strictly positive real constant. We propose a single-run optimization algorithm as follows, with $\hat{Y}(n)$ denoting an estimate of $Y(n,x)$ generated while the optimization algorithm is running. The algorithm is similar to the one in [Meketon 1987].

Algorithm 2. A single-run optimization algorithm

(1) Initialize:
Choose a sequence of numbers $\{a(n) = K/n, n = 1, 2, \ldots\}$.

$$\hat{Y}(0) = 0, \quad n = 1, \quad x(n) = \mathrm{unif}[-u, u]$$

(2) Simulate: Generate $\epsilon(n) \sim \mathrm{Normal}(\mu, \sigma^2)$.

$$\hat{Y}(n) = x(n) \cdot \hat{Y}(n-1) + \epsilon(n)$$

(3) Update:

$$\tilde{x}(n+1) = x(n) - a(n)[\hat{Y}(n) - \alpha]$$

$$x(n+1) = \pi(\tilde{x}(n+1))$$

$$n = n + 1,$$

where π ($= \pi_1$ or π_2 defined respectively by (9) or (10) below) denotes a projection operator into the set Γ.

(4) If the chosen stopping criterion is satisfied, stop. $x(n)$ is an estimate of x^*. Otherwise, go to step 2. [End of algorithm]

It should be emphasized that in step 2 of Algorithm 2, updating of $x(n)$ is done after observing only one value of $\hat{Y}(n)$, a noisy, biased, and correlated sample of $M(x(n))$. The updating scheme belongs to the Robbins-Monro type. After updating, simulation is continued with the old state but the new parameter value.

The projection operator π in step 3 is to ensure that $x(n+1) \in \Gamma$. Writing $\tilde{x}(n+1)$ as $\tilde{x}$ and the logical operator "and" as $\wedge$, we define two different projection operators π_1 and π_2 by

$$\pi_1(\tilde{x}) = \begin{cases} \tilde{x}, & -1 < \tilde{x} < 1 \\ \text{unif}[-u,u], & \text{otherwise} \end{cases} \tag{9}$$

$$\pi_2(\tilde{x}) = \begin{cases} \tilde{x}, & -1 < \tilde{x} < 1 \\ \text{unif}[x(n),u], & (1 \le \tilde{x}) \wedge (-u \le x(n) \le u) \\ \text{unif}[2u - x(n),u], & (1 \le \tilde{x}) \wedge (u < x(n)) \\ \text{unif}[-u,u], & (1 \le \tilde{x}) \wedge (x(n) < -u) \\ \text{unif}[-u,x(n)], & (\tilde{x} \le -1) \wedge (-u \le x(n) \le u) \\ \text{unif}[-u,-2u - x(n)], & (\tilde{x} \le -1) \wedge (x(n) < -u) \\ \text{unif}[-u,u], & (\tilde{x} \le -1) \wedge (u < x(n)) \end{cases} \tag{10}$$

π_1 is a simple projector so that the algorithm restarts from a random point within the interval $[-u,u]$ when $\tilde{x}$ is outside Γ. π_2 is a more sophisticated projector using some current information (namely $x(n)$) of the system. Basically, π_2 works as follows. If $\tilde{x} > 1$, then a random step is taken from the current $x(n)$ towards u. Because of a technical condition required for convergence (see [Leung 1990]), it is necessary to always project $\tilde{x}$ onto a compact set ($[-u,u]$ here). This is the reason why we need to consider separately the cases when $\tilde{x} > 1$ and $x(n)$ itself is outside $[-u,u]$ (but within [-1,1]), as shown in the third and fourth lines in (10). The case of $\tilde{x} < 1$ is similar.

It should be noted that these two projection operators are by no means optimal in any sense. The objective of using two different operators is to study the effect of projection on the behavior of the optimization algorithm. π_1 and π_2 represent somewhat arbitrary choices for a simple-minded and a reasonably "intelligent" projection scheme.

As before, we assume that a suitable stopping criterion has been chosen in step 4.

3.3 Finite-Time Behavior of the Algorithm

Consider the problem defined by (6) and (8) under assumptions (A3.1.1) to (A3.1.3). We can derive a set of equations, recursive in n, which describes the finite-time behavior of Algorithm 2 in terms of the distribution function of $x(n)$. In principle, solving the equations leads to an exact solution for the distribution function of $x(n)$. In practice, these equations have to be enumerated using numerical methods.

The detailed mathematical derivation is of considerable length (see [Leung 1990]) and is omitted here. Instead, we give an example of the results from enumerating the equations. This example serves to illustrate the performance of Algorithm 2 under the two alternative projection operators π_1 and π_2.

For convenience, we shall use a single index of the performance of the algorithms, the percentage root mean squared error denoted by % x-RMSE(n). It is defined as

$$\% \ x\text{-RMSE}(n) = 100 \cdot \{\text{MSE}[x(n)]\}^{1/2}/x^*.$$

Recall that

$$\text{MSE}[x(n)] = \text{E}\left[(x(n) - x^*)^2\right].$$

Using the set of recursive equations, we can numerically obtain, for $n = 1, 2, \ldots$, the distribution of $x(n)$ and hence the expectation on the RHS of the above equation.

To evaluate the finite-time behavior of Algorithm 2 in general terms, we compare it with a standard simulation (without optimization) of the AR(1) process at a fixed parameter x, as defined in (6). Suppose we are interested in estimating the steady-state performance measure

$$M(x) = \lim_{n\to\infty} \text{E}[Y(n,x)] = \mu/(1-x).$$

A standard point estimator for $M(x)$ over one simulation run of n observations is

$$\hat{M}(x,n) = \frac{1}{n}\sum_{i=1}^{n} Y(i,x).$$

Tedious algebraic calculation gives the mean squared error of this estimator as
$\text{MSE}[\hat{M}(x,n)] =$

$$\frac{\sigma^2}{n(1-x)^2} - \frac{(1-x^n)}{n^2(1-x)^3}\left[\frac{\sigma^2(2x + x^2 - x^{n+2})}{1+x} - \frac{\mu^2 x^2(1-x^n)}{1-x}\right].$$

For comparison of this steady-state estimation problem with the optimization algorithm, we fix $x = x^*$ and define the percentage root mean squared error of $\hat{M}(x^*,n)$ by

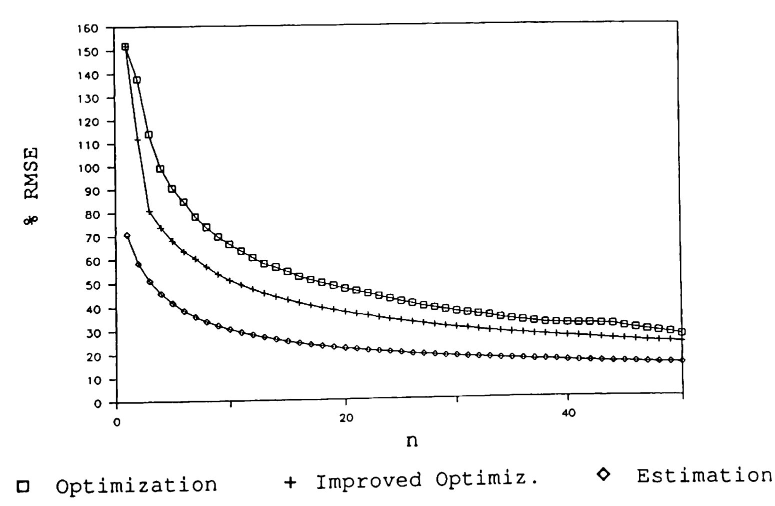

Figure 2. Behavior of Algorithm 2 when $\alpha = 2$

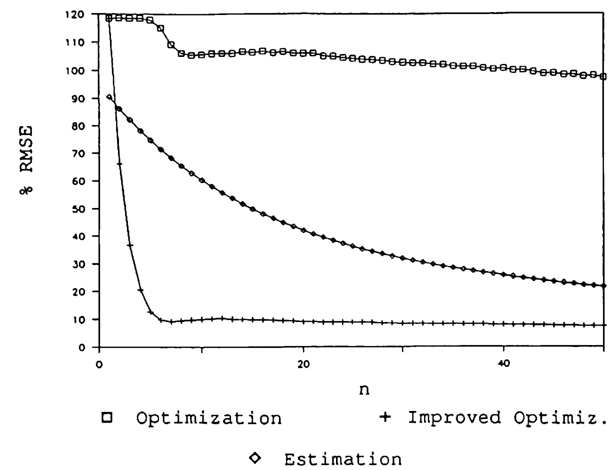

Figure 3. Behavior of Algorithm 2 when $\alpha = 10$

$$\% \, M\text{-RMSE}(n) = 100 \cdot \left\{ \text{MSE}\left[\hat{M}(x^*, n)\right] \right\}^{1/2} / M(x^*).$$

This quantity is chosen to investigate the difference between the rate of $x(n)$ converging to x^* and the rate of $\hat{M}(x^*, n)$ reaching its steady state value in a standard simulation.

Now we present some numerical results of the percentage root mean squared errors for both the optimization case (Algorithm 2) and the estimation case (conventional simulation). Parameters used are: $\alpha = 2$ (hence $x^* = 0.5$) or 10 ($x^* = 0.9$), $K = 1, u = 0.99, \mu = 1, \sigma^2 = 1$. Figures 2 and 3 show the values of % x-RMSE(n) and % M-RMSE(n) for $n = 1, \ldots, 50$. In the figures, Algorithm 2 with projection operator π_1 is labelled as "optimization"; Algorithm 2 with π_2 is labelled as "improved optimization" and estimation of $M(x^*)$ using conventional simulation is labelled as "estimation". The vertical axis, labelled as % RMSE, represents the % x-RMSE for the optimization algorithms and the % M-RMSE for steady-state estimation.

From Figure 2 we see that the estimator of the optimizer, $x(n)$, generated by Algorithm 2, converges roughly as fast as the estimator of $Y(x^*)$ does in a conventional simulation. The difference between π_1 and π_2 is not very pronounced; nevertheless π_2 shows a slightly faster initial convergence rate. Figure 3 clearly suggests the superiority of π_2. It helps the algorithm converge quickly to a small neighborhood of x^*. It is interesting to note that in this latter case, the optimization algorithm with π_2 converges even faster than the steady-state estimator in a standard simulation.

From these numerical results, it seems worthwhile to use the more complicated projection operator π_2. The extra computation required by it is most likely to be more than offset by the faster initial convergence rate offered. At the same time, we see that the (finite-time) convergence rate of the single-run optimization algorithm is very fast, comparable to the rate of a standard simulation reaching steady state. This analysis complements the experimental evidence obtained earlier [Suri and Zazanis 1988, L'Ecuyer et al. 1989, Suri and Leung 1989] that single-run optimization algorithms of this type are very efficient.

4. CONCLUSIONS

We have analyzed the finite-time behavior of two simulation optimization algorithms. Results suggest that the Robbins-Monro procedure when applied in a conventional way can be slow; however it is very fast when applied in a single-run optimization algorithm. This work also demonstrates the importance of understanding the finite-time behavior of, and the impact of the projection operator on, such simulation optimization algorithms. Although the results are developed for two simple systems, they provide some insight into the performance of the Robbins-Monro procedure in the simulation environment.

REFERENCES

Chung, K.L. (1954), "On a Stochastic Approximation Method", *Annals of Mathematical Statistics 25*, 463-483.

Farrell, W. (1977), "Literature Review and Bibliography of Simulation Optimization", *Proceedings of the 1977 Winter Simulation Conference*, 117-124.

Fishman, G.S. (1972), "Bias Considerations in Simulation Experiments", *Operations Research 20*, 785-790.

Glynn, P.W. (1986), "Optimization of Stochastic Systems", *Proceedings of the 1986 Winter Simulation Conference*, 52-59.

Hooke, R. and T.A. Jeeves (1961), "A Direct Search Solution of Numerical and Statistical Problems", *Journal of the Association for Computing Machinery 8*, 2, 212-229.

Jacobson, S.H. and L.W. Schruben (1989), "Techniques for Simulation Response Optimization", *Operations Research Letters 8*, 1, 1-9.

Kelton, W.D. (1986), "Replication Splitting and Variance for Simulating Discrete-Parameter Stochastic Processes", *Operations Research Letters 4*, 6, 275-279.

Kelton, W.D. and A.M. Law (1984), "An Analytical Evaluation of Alternative Strategies in Steady-State Simulation", *Operations Research 32*, 1, 169-184.

Kushner, H.J. and D.S. Clark (1978), *Stochastic Approximation Methods for Constrained and Unconstrained Systems*, Springer-Verlag, New York, NY.

Kushner, H.J. and H. Huang (1979), "Rates of Convergence for Stochastic Approximation Type Algorithms", *Society for Industrial and Applied Mathematics Journal on Control and Optimization 17*, 5, 607-617.

L'Ecuyer, P., N. Giroux, and P.W. Glynn (1989), "Stochastic Optimization by Simulation: Some Experiments with a Simple Steady-State Queue", Technical Report No. 31, Department of Operations Research, Stanford University, Stanford, CA.

Leung, Y.T. (1990), "Single-Run Optimization of Discrete-Event Simulations", Ph.D. dissertation, Department of Industrial Engineering, University of Wisconsin - Madison, Madison, WI.

Ljung, L. (1977), "Analysis of Recursive Stochastic Algorithms", *Institute of Electrical and Electronic Engineers Transactions on Automatic Control AC-22*, 4, 551-575.

Major, P. and P. Révész (1973), "A Limit Theorem for the Robbins-Monro Approximation", *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete 27*, 79-86.

Meketon, M.S. (1983), "A Tutorial on Optimization in Simulations", presented at the 1983 Winter Simulation Conference.

Meketon, M.S. (1987), "Optimization in Simulation: A Survey of Recent Results", *Proceedings of the 1987 Winter Simulation Conference*, 58-67.

Robbins, H. and S. Monro (1951), "Stochastic Approximation Method", *Annals of Mathematical Statistics 22*, 400-407.

Sacks, J. (1958), "Asymptotic Distribution of Stochastic Approximation Procedures", *Annals of Mathematical Statistics 29*, 373-405.

Stroup, D.F. and H.I. Braun (1982), "On a New Stopping Rule for Stochastic Approximation", *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete 60*, 535-554.

Suri, R. and Y.T. Leung (1989), "Single Run Optimization of Discrete Event Simulations: An Empirical Study Using the M/M/1 Queue", *Institute of Industrial Engineers Transactions 21*, 1, 35-49. Also as Technical Report No. 87-3, Department of Industrial Engineering, University of Wisconsin - Madison, Madison, WI, 1987.

Suri, R. and M.A. Zazanis (1988), "Perturbation Analysis Gives Strongly Consistent Sensitivity Estimates for the M/G/1 Queue", *Management Science 34*, 1, 39-64.

Turnquist, M.A. and J.M. Sussman (1977), "Toward Guidelines for Designing Experiments in Queueing Simulation", *Simulation 28*, 137-144.

Wardi, Y. (1988), "Simulation-Based Stochastic Algorithm for Optimizing GI/G/1 Queues", working paper, Department of Industrial Engineering, Ben Gurion University of the Negev, Beer Sheva, Israel.