Feature Article

Optimization for Simulation: Theory vs. Practice

Michael C. Fu
Robert H. Smith School of Business and Institute for Systems Research, University of Maryland, College Park, Maryland 20742-1815, mfu@rhsmith.umd.edu

Probably one of the most successful interfaces between operations research and computer science has been the development of discrete-event simulation software. The recent integration of optimization techniques into simulation practice, specifically into commercial software, has become nearly ubiquitous, as most discrete-event simulation packages now include some form of "optimization" routine. The main thesis of this article, however, is that there is a disconnect between research in simulation optimization—which has addressed the stochastic nature of discrete-event simulation by concentrating on theoretical results of convergence and specialized algorithms that are mathematically elegant—and the recent software developments, which implement very general algorithms adopted from techniques in the deterministic optimization metaheuristic literature (e.g., genetic algorithms, tabu search, artificial neural networks). A tutorial exposition that summarizes the approaches found in the research literature is included, as well as a discussion contrasting these approaches with the algorithms implemented in commercial software. The article concludes with the author's speculations on promising research areas and possible future directions in practice.
(*Simulation Optimization; Simulation Software; Stochastic Approximation; Metaheuristics*)

1. Introduction

Until the end of the last millennium, optimization and simulation were kept pretty much separate in practice, even though there was a large body of research literature relevant to combining them. In the last decade, however, "optimization" routines (the reason for the quotes will be explained shortly) have prominently worked their way into simulation packages. That this is a fairly recent development is revealed by the fact that all of the software routines for performing simulation optimization listed in the current edition of Law and Kelton (2000, p. 664, Table 12.11)—AutoStat, OptQuest, OPTIMIZ, SimRunner,

0899-1499/02/1403/0192$5.00
1526-5528 electronic ISSN

Table 1 Optimization for Simulation: Commercial Software Packages

Optimization Package (Simulation Platform)	Vendor (URL)	Primary Search Strategies
AutoStat (AutoMod)	AutoSimulations, Inc. (www.autosim.com)	evolutionary, genetic algorithms
OptQuest (Arena, Crystal Ball, et al.)	Optimization Technologies, Inc. (www.opttek.com)	scatter search and tabu search, neural networks
OPTIMIZ (SIMUL8)	Visual Thinking International Ltd. (www.simul8.com)	neural networks
SimRunner (ProModel)	PROMODEL Corp. (www.promodel.com)	evolutionary, genetic algorithms
Optimizer (WITNESS)	Lanner Group, Inc. (www.lanner.com/corporate)	simulated annealing, tabu search

and WITNESS Optimizer (shown in Table 1)—were not in existence at the time of the earlier printings. The goal of these routines is to seek improved settings of user-selected system parameters with respect to the performance measure(s) of interest, but contrary to the use of mathematical programming software packages, the user has no way of knowing if an optimum has actually been reached (hence the quotations around optimization at the beginning of this paragraph). Like so many other developments in the OR/CS interface, this has only become practical with the immense leaps in computational power, which have greatly benefited both optimization and simulation. For optimization, this has led to the solution of large-scale decision-making problems in the real world, whereas for simulation, it has meant that entire complex systems could be realistically modeled to the point of providing useful operational and managerial decision support. It used to be that for a realistic system of interest (e.g., a manufacturing plant), estimation by itself (perhaps with some basic sensitivity analysis) all but expended the simulation "budget" in terms of computing time (hours or days), so that performing optimization was unthinkable, because it would require at least another order of magnitude of computational resources. Now these "optimization" routines can be performed on PCs in roughly the same amount of time as estimation required previously. This, however, may still mean days:

> Optimization analyses take a large number of runs. You can use AutoStat to make runs on multiple machines on your network… You can take advantage of other machines to make runs overnight or on weekends (Bitron 2000).

Here are some important pieces of evidence indicative of the new marriage between optimization and simulation in practice.

- At present, nearly every commercial discrete-event simulation software package contains a module that performs some sort of "optimization" rather than just pure statistical estimation. Contrast this with the status in 1990, when none of the packages included such an option.
- The most recent editions of two widely used discrete-event simulation textbooks, Law and Kelton (2000) ("used by more than 70,000 people worldwide!" screams the cover of the March–May 2001 brochure announcement of Simulation Short Courses given by first author) and Banks et al. (2000) have added new sections (12.6 and 12.4, respectively) dedicated to the topic.
- The term "simulation optimization" has itself become more widespread; for example, it is one of the new entries in the updated second edition of the *Encyclopedia of Operations Research and Management Science* (Gass and Harris 2000).

The first question one might ask is, "Why can't one just pop the simulation box into one of the existing optimization packages?" That is basically the philosophy behind the so-called sample path optimization approach, to be described later. On the other

hand, here is the counter view of one of the software providers (www.opttek.com, November 2000):

> The most commonly used optimization procedures—linear programming, nonlinear programming and (mixed) integer programming—require an explicit mathematical formulation. Such a formulation is generally impossible for problems where simulation is relevant, which are characteristically the types of problems that arise in practical applications.

The term "simulation" will henceforth be shorthand for stochastic discrete-event simulation, meaning that the random nature of the system will be implicitly understood and the underlying models are discrete-event systems such as queueing networks. In fact, it is the stochastic nature that is key in all of the discussion, and one central thesis of this article is that the currently implemented optimization algorithms do not adequately address this characteristic. The focus on discrete-event simulation has two rationales: It is the primary domain of operations researchers in stochastic simulation (as opposed to, for example, stochastic differential equations in the fields of computational finance or stochastic control), and it is where optimization and simulation have come together most prominently in commercial software. The primary application areas are manufacturing, computer and communications networks, and business processes.

The selection of the title of this article, "Optimization *for* Simulation," was made quite deliberately. The two most recent comprehensive survey articles on the subject, Fu (1994) and Andradóttir (1998), are titled "Optimization via Simulation" and "Simulation Optimization," respectively, reflecting the two terms most commonly used in the field (see also Swisher et al. 2001). These two titles more accurately reflect the state of the art in the research literature, whereas the purpose of this article is to explore the linkages (and lack thereof) with the practice of discrete-event simulation. In that sense, it is not an equal partnership but a subservient one, in which the optimization routine is an add-on to the underlying simulation engine, as depicted in Figure 1. In contrast, one can view the recent developments in stochastic programming as the converse, simulation for optimization, as depicted in Figure 2, where Monte Carlo simulation is the add-on used to generate scenarios for math programming

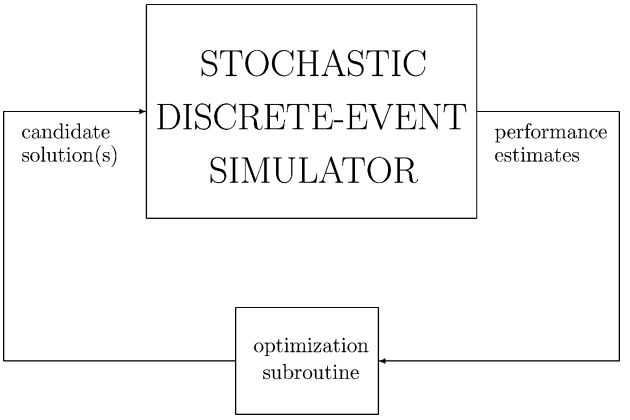

Figure 1 Optimization for Simulation: Commercial Software

formulations from a relatively small underlying set of possible realizations. One of the primary application areas of this approach is financial engineering, e.g., portfolio management.

In the literature, there is a wide variety of terms used in referring to the inputs and outputs of a simulation optimization problem. Inputs are called (controllable) parameter settings, values, variables, (proposed) solutions, designs, configurations, or factors (in design of experiments terminology). Outputs are called performance measures, criteria, or responses (in design of experiments terminology). Some of the outputs are used to form an objective function, and there is a constraint set on the inputs. Following deterministic optimization common

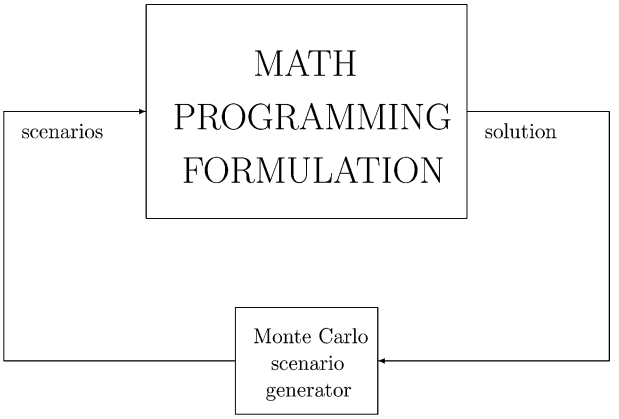

Figure 2 Simulation for Optimization: Stochastic Programming

usage, we will use the terms "variables" and "objective function" in this article, with the latter comprised of performance measures estimated from simulation (consistent with discrete-event simulation common usage). A particular setting of the variables will be called either a "configuration" or a "design."

The general setting of this article is to find a configuration or design that minimizes the objective function:

$$\min_{\theta \in \Theta} J(\theta) = E[L(\theta, \omega)], \tag{1}$$

where $\theta \in \Theta$ represents the (vector of) input variables, $J(\theta)$ is the objective function, ω represents a sample path (simulation replication), and L is the sample performance measure. We will use $\widehat{J}$ to represent an estimate for $J(\theta)$, e.g., $L(\theta, \omega)$ would provide one such estimator that is unbiased. The constraint set Θ may be either explicitly given or implicitly defined. For simplicity in exposition, we assume throughout that the minimum exists and is finite, e.g., Θ is compact or finite, as opposed to using "inf" and allowing an infinite value. Throughout, J will be scalar and an expectation; multiple performance measures can be handled simply by assigning appropriate weights and combining to form a single objective function, though this may not always be desirable or practical (but very little has been done on multi-response simulation optimization). Note that probabilities can be handled as expectations of indicator functions, but that quantiles (e.g., the median) and measures such as "most likely to be the best" (e.g., mode) are excluded by this form of performance measure. Most of the commercial software packages also allow the practically useful extension of the setting in (1) to that of including explicit inequality constraints on output performance measures (as opposed to the indirect way of incorporating them into the objective function by way of a penalty function or Lagrange multipliers).

The categories of inputs are generally divided into two types: qualitative and quantitative. The former are characterized by not having a natural ordering (either partial or full). The latter are then further divided into two distinct domains of discrete and continuous variables, analogous to deterministic optimization, where the approaches to these types of problems can also be quite different.

Real-World Example: Call Center Design and Operations

Customer Relationship Management (CRM) is currently one of the hottest topics (and buzzwords) in business management (ranked #1 in technology trends for 2001 by M. Vizard, the Editor in Chief of *Info World*, p. 59 of January 8, 2001 issue, "Top 10 technology trends for 2001 all ask one thing: Are you experienced?").

> This isn't just about providing adequate support when a customer needs help, but rather about offering the customer an overall relationship with the company that's valuable, compelling and unavailable anywhere else… (*The Industry Standard*, "You and Your Customer," Nov. 6, 2000, pp. 154–155.)

For example, IBM has an "eCare" program, with the objective of delighting the customer, which translates into personalizing information and support on the Web. One of Oracle's major advertising campaigns in 2001 promises "global CRM in 90 days." *Amazon.com* has a Vice-President of CRM, though the acronym has a little different twist on it: Customer-Relationship Magic. The technology underlying CRM involves data warehousing and data mining, but for many businesses, the key CRM storefront is the call center that handles customer orders (for products or services), inquiries, requests, and complaints. "Customer support, in fact, is one of the most common ways businesses can put their CRM strategy to work" (ibid.). Far from being just a staid telephone switchboard, the state of the art integrates traditional call operations with both automated response systems (computer account access) and Internet (Web-based) services and is often spread over multiple geographically separate sites. For this reason, the term "call center" is rapidly being supplanted by the more dynamic and all-encompassing appellation "contact center" to reflect more accurately the evolving nature of the diverse activities being handled.

Most of these centers now handle multiple sources of jobs ("multichannel contact"), e.g., voice, e-mail, fax, interactive Web, which require different levels of operator (call agent) training, as well as different priorities, as voice almost always preempts any of the other contact types (except possibly interactive Web). There are also different types of jobs according to the

service required, e.g., an address change versus checking account balance versus a more involved transaction or request; hence, the proliferation of bewildering menu choices on the phone. Furthermore, because of individual customer segmentation, there are different classes of customers in terms of priority levels. In particular, many call centers have become quite sophisticated in their routing of incoming calls by differentiating between preferred, ordinary, and "undesirable" (those that actually cost more to serve than their value added) customers. The easiest way to implement this is to give special telephone numbers to select customers. Most airline frequent flier programs do this, so on Continental Airlines, I am special and pampered, but on US Airways I am ordinary. Other call center systems request an account number and use this as part of their routing algorithm. When I punch in my account number to Charles Schwab's Signature Service line, I receive an operator almost immediately. "Based on a customer's code, call centers route customers to different queues. Big spenders are whisked to high-level problem solvers. Others may never speak to a live person at all..." (*Business Week* Cover Story, pp. 118–128, October 23, 2000) "Walker Digital, the research lab run by Priceline founder Jay S. Walker, has patented a 'value-based queuing' of phone calls that allows companies to prioritize calls according to what each person will pay. As Walker Digital CEO Vikas Kapoor argues, customers can say: 'I don't want to wait in line—I'll pay to reduce my wait time'" (ibid.). What this means is that call-routing algorithms (implemented as rules in the automatic call distributor—ACD) are now a key integral part of providing the right level of customer service to the right customers at the right time.

Designing and operating such a call center (the CTI—computer telephony integration—strategy) entails many stochastic optimization problems that require selection of optimal settings of certain variables, which may include quantitative (e.g., number of operators at each skill level and for each particular class of customer, number and types of telecommunications devices) and qualitative (e.g., what routing algorithm and type of queue discipline to use: FCFS, priority to elite customers, or something else) dimensions. The objective function will consist of metrics associated with both customers and agents. For example, there are cost components associated with service level performance measures such as waiting times (most commonly the mean or the probability of waiting more than a certain amount of time, possibly weighted by class type) and operational costs associated with agent wages and network usage (trunk utilization). Abandonment rates of waiting customers, percentage of blocked calls (those customers that receive a busy signal), and agent utilization are other factors that are considered. Clearly, there is a trade-off that must be made between customer service levels and the cost of providing service. As in any optimization problem, this could be expressed with a single objective function, e.g., minimize costs subject to a number of different constraints, such as pre-specified customer service levels for each class of customer (lower bound for preferred customers, perhaps upper bound for undesirable ones?).

Toy Example: Single-Server Queue

The most studied OR model in the illustrious history of queueing theory still necessitates simulation in many cases. Consider a first-come, first-served, single-class, single-server queue with unlimited waiting room, such that customer service times are drawn independently from the same probability distribution (single class of customers) and the controllable variable is the speed of the server. Let θ denote the mean service time of the server (so $1/\theta$ corresponds to the server speed). Then one well-studied optimization problem uses the following objective function:

$$J(\theta) = E[W(\theta)] + c/\theta, \tag{2}$$

where W is the steady-state time spent in the system and c is the cost factor for the server speed. In other words, a higher-skilled worker costs more. Since W is increasing in θ, the objective function clearly quantifies the trade-off between customer service level and cost of providing service. This could be viewed as the simplest possible case of the call center design problem, where there is a single operator whose skill level must be selected. Simulation of this system requires specification of the arrival process and the service time distribution. In addition to its honored place in queueing theory, this model is often the first system

used in textbooks to illustrate discrete-event simulation (e.g., Law and Kelton 2000). For the simplest $M/M/1$ queue in steady state, this problem is analytically tractable, and thus has served as an easy test case for optimization procedures, especially those based on stochastic approximation.

Another Academic Example: (s, S) Inventory Control System

This is another well-known OR model from inventory theory, in which the two parameters to be optimized, s and S, correspond to the re-order level and order-up-to level, respectively. When the inventory level falls below s, an order is placed for an amount that would bring the current level back up to S. Optimization is generally carried out by minimizing a total discounted or average cost function—consisting of ordering, holding, and backlogging or lost sales components—or just ordering and holding costs but subject to a service level constraint. In research papers on simulation optimization, this example is nice, because it is the simplest multidimensional problem (as opposed to the previous scalar one), and being in two dimensions, it has a nice graphical representation in search procedures (see Section 4). Thus, it has been used as a test case for nearly all the procedures in the research literature discussed in Section 4, i.e., stochastic approximation, sequential response surface methodology, retrospective optimization (an early incarnation of sample path optimization), statistical ranking and selection, and multiple comparisons.

To attack the generic problem posed by (1), the five packages listed in the opening paragraph use metaheuristics from combinatorial optimization based on evolution strategies such as genetic algorithms, tabu search, and scatter search (see Glover et al. 1999), with some adaptation of other techniques taken from the deterministic optimization literature, e.g., neural networks and simulated annealing (even though the latter is probabilistic in nature, it has been primarily applied to deterministic problems). On the other hand, the research literature in simulation optimization (refer to Andradóttir 1998 or Fu 1994) is dominated by continuous-variable stochastic approximation methods and random search methods for discrete-variable problems, which consist primarily of search strategies iterating a single point, versus the group or family of points adopted by the metaheuristics above. The continuous-variable algorithms are predominantly based on local gradient search.

Thus, in terms of software implementation, the available routines are based on approaches outside of the *simulation* research literature. Indeed, other than in the Winter Simulation Conference Proceedings, one would be hard-pressed to find published examples of metaheuristics represented in archival journals on simulation. "Why is this the case?" you might ask. There appear to be two major barriers: Either the algorithms that are implemented are not *provably* convergent, or the use of simulation is secondary. In the latter case, it seems more appropriate that the algorithm be published in the *Journal of Heuristics*—with roots in the combinatorial optimization community—than in the *ACM Transactions on Modeling and Computer Simulation*, the most highly respected OR journal dedicated to stochastic simulation, whose founding editor is from a computer science department but whose editorial board is dominated by OR researchers in the simulation community.

"What will the remainder of this article try to accomplish?" you might naturally ask at this point. (Or, "Why should I read any further, since I already have the main idea?") It will attempt to do the following:

- Explain why optimization for simulation should not merely consist of deterministic algorithms applied to a black box that happens to be a simulation model.
- Provide a representative, but by no means exhaustive, high-level description of the algorithms and theoretical convergence results in the simulation optimization literature and of the relevant related results from the stochastic optimization literature.
- Contrast with the routines that are found in commercial discrete-event simulation software by describing the general search strategies of two packages and delving into the specific user-specified parameters and provided outputs for one of them.
- Touch upon research directions that are important or promising, in the author's humble opinion, and speculate on the future of optimization for simulation, both in theory and in practice.

The remainder of this article is organized as follows. Section 2 expatiates further on those features that make optimization for simulation more than simply a straightforward implementation of deterministic algorithms. It includes a summary of challenges common to both research and practice, along with key issues that separate the two. A (very) brief tutorial on simulation output analysis and probabilistic convergence modes is also provided as background or review material. Section 3 surveys the research approaches for simulation optimization and provides a flavor of the theoretical results in the literature. Section 4 contrasts research with practice by describing two commercial software routines that implement optimization for simulation. Future directions in research and in practice are then discussed in Section 5. The article concludes with a brief section on sources for probing further.

2. What Makes Simulation Optimization Different?

As alluded to earlier, what makes simulation optimization doubly difficult on top of the ordinary deterministic optimization setting is its stochastic nature. A nice summary of this key difficulty is provided by Banks et al. (2000, p. 488):

> Even when there is no uncertainty, optimization can be very difficult if the number of design variables is large, the problem contains a diverse collection of design variable types, and little is known about the structure of the performance function. Optimization via simulation adds an additional complication because the performance of a particular design cannot be evaluated exactly, but instead must be estimated. Because we have estimates, it may not be possible to conclusively determine if one design is better than another, frustrating optimization algorithms that try to move in improving directions. In principle, one can eliminate this complication by making so many replications, or such long runs, at each design point that the performance estimate has essentially no variance. In practice, this could mean that very few alternative designs will be explored due to the time required to simulate each one.

In the problem setting of (1), the usual *goals* of optimization can be stated succinctly as follows:

(a) Finding $\arg\min_{\theta\in\Theta} J(\theta)$

(or at least one element, if the arg min is a set, i.e., the problem has multiple optima).

(b) Returning $\min_{\theta\in\Theta} J(\theta)$.

For example, think of the practical problem of finding the quickest route to work each morning. Then the corresponding questions to be answered are: (a) Which roads should I take? (b) How long will it take? In deterministic optimization, all the emphasis is on (a), because (b) is trivial once (a) is accomplished, i.e., if $\theta^* \in \arg\min J(\theta)$, then $\min_{\theta\in\Theta} J(\theta) = J(\theta^*)$. In other words, one does not generally distinguish between the two as being separate problems in the deterministic setting. In a stochastic (and real-life) setting, however, one must change (b) to the following: *estimating* $\min_{\theta\in\Theta} J(\theta)$. In fact, sometimes it is only (a) that is of ultimate (or primary) interest, and J is simply the means to the end. This is usually the case in the going-to-work example, since, unless you are cutting it very close to meet a tight scheduled appointment, you are probably more interested in finding the quickest route than in precisely estimating the actual total travel time. Furthermore, I "know" that in general taking the beltway is much quicker than taking all local roads, but I have only a rough estimate of the time it takes for the portion of time spent on the highway (10 to 15 minutes), and very little idea as to how long a totally local route would take (at least an hour?). Here are some other examples:

- Preventive Maintenance: Finding an optimal (or a good) policy is most likely the goal, with the cost estimates often only a rough gauge of various operational aspects.
- Manufacturing Plant Design: Selection of the best design is the primary goal, rather than the cost (or profit) estimate.
- Derivatives Pricing: Options with early exercise opportunities require the determination of an optimal policy in order to find the price; however, in this case, the situation is reversed, as it is the estimated price that is paramount and not the policy itself.

The actual *process* of optimization can be divided into two parts:

(I) Generating candidate solutions.

(II) Evaluating solutions.

In optimization for simulation, the perspective of practice in terms of coming up with algorithms is to concentrate on the first step, just as in the deterministic case, treating the simulation model essentially

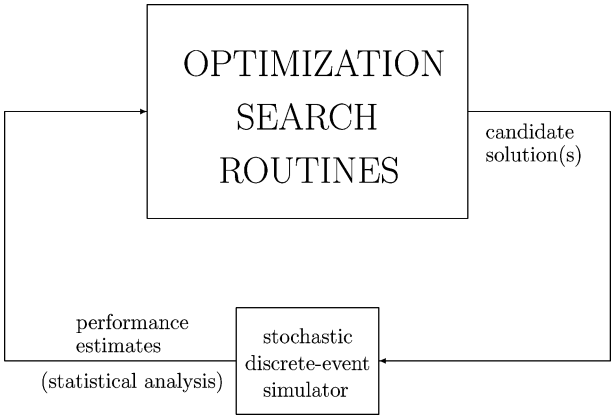

Figure 3 Optimization for Simulation: Practice Perspective

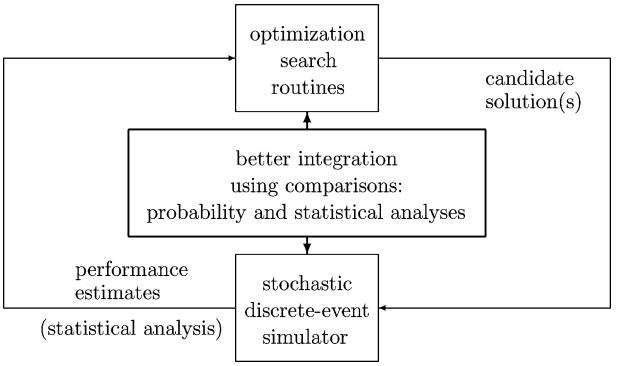

Figure 4 Optimization for Simulation: Future Needs

as just another function generator with some statistical analysis, as shown in Figure 3. In optimization for simulation, however, most of the computation is expended in estimating $J(\theta)$ for values of θ in the search, a reversal of the deterministic setting, where the search is the primary computational burden. Thus, the commercial software view shown in Figure 1, where optimization is viewed as simply another subroutine add-on, also reflects the computational balance between the two functions. A major determinant of the computational cost for a simulation optimization algorithm is the number of simulation replications used to estimate $J(\theta)$ for each θ. For example, there is no reason a priori to assume that the number of replications should be the same for all values of θ nor the same for each iteration in the search process. In sum, a key feature that is not a factor in deterministic settings is the trade-off between the amount of computational effort needed to estimate the performance at a particular solution versus the effort in finding improved solution points (or families). A related (motivating?) point is that the focus should therefore be on *comparing relative* performance instead of estimating absolute performance, i.e., order is the essential goal during the search process. In contrast, when there is an absence of randomness, calculating absolute performance is essentially indistinguishable from comparing relative performance.

To summarize, the process in a stochastic setting should be modified as follows (shown in Figure 4):

(I) Iterative but Integrated: Searching and Comparing; finding θ^*.

(II) Final: Estimating the optimal value of objective function $J(\theta^*)$.

As noted earlier, step (II) may or may not be the ultimate objective.

Because, in the deterministic setting, evaluating and comparing are considered essentially the same step, currently implemented simulation optimization routines do not really address (much less exploit) the notion of ordinal comparisons. To reiterate this crucial point:

> It is generally easier to compare solutions and find relative ordering among them than it is to estimate them precisely.

This is the main basis of the so-called ordinal optimization approach, where the goal is approximate order, rather than precise values. As an aside, this is also the philosophy behind the analytic hierarchy process (AHP), where relative weights are considered key.

Banks et al. (2000, pp. 488–489) break down the approaches toward optimization via simulation into four categories:

- guarantee asymptotic convergence to the optimum (generally for continuous-valued parameters);
- guarantee optimality under deterministic counterpart (i.e., if there were no statistical error or sampling variability; generally based on mathematical programming formulations);
- guarantee a prespecified probability of correct selection (generally from a prespecified set of alternatives);

• robust heuristics (generally based on combinatorial search algorithms that use an evolutionary approach, e.g., genetic algorithms).

Going down the list, there is a transition from complete confidence in optimality, albeit in an unrealizable context, to workable solutions that apply in practical settings. This mirrors the tug of war between research and practice that involves a dueling between approaches that provide quick rough-and-ready solutions with no performance guarantees (based on heuristics, with no statistical analysis whatsoever) versus the more rigorous mathematical approaches dominating the academic literature that either guarantee convergence or probability of correct selection.

Here is an illustration of each of these using the toy example, where the objective is to select the skill level (speed) of the call center operator.

• The operator skill level is variable over a range of values, which can be either continuous (e.g., for stochastic approximation algorithms) or discrete (e.g., using random search). The algorithm will find an optimum (single value) for sure (100% confidence) if the algorithm is run "long enough."

• Again, the operator skill level is variable over a continuous range of values. The algorithm returns an optimum (single value) under the case where *each simulation* (versus the algorithm itself, in the previous case) is run "long enough."

• Reduce the problem to a fixed (relatively small) finite number of operator skill levels from which to choose. The algorithm will select a skill level with objective function value within ϵ of the best at a $(1-\alpha)100\%$ confidence level, where the confidence level is generally a *lower* bound.

• The operator skill level is variable over a range of values, which could be discrete or continuous. The routine will return the best found values from a family and provide an estimate of improvement from the beginning of the search.

Parallel results could be described for the inventory example, corresponding to modeling demand and inventory quantities as continuous valued (e.g., gallons of oil), or discrete valued (e.g., number of books).

2.1. Research and Practice: Key Issues

It can be argued that both research and practice have adapted approaches from deterministic optimization, as the summary in Table 2 can be used to demonstrate. Gradient search, random search, and math programming adaptations dominate the research literature, whereas software implementations incorporate one or more of the other approaches. Key issues separating (or facing both) research and practice include the following:

Table 2 Approaches from Deterministic Optimization

Approach	Key Features
Gradient search	Move locally in most promising direction, according to gradient
Random search	Move randomly to new point, no information used in search
Simulated annealing	Sometimes move in locally worse directions, to avoid being trapped in local extrema
Genetic algorithms and scatter search	Population based, generates new members using (local) operations on attributes of current members
Tabu search	Use memory (search history) to avoid tabu moves
Neural networks	(Nonlinear) function approximation
Math programming	Powerful arsenal of rigorously tested software

• stochastic comparisons;

• family of solutions versus a single point, and use of memory;

• continuous versus discrete;

• convergence and statistical validity.

The first issue has already been discussed, so the rest of the discussion in this section will touch on the remaining issues.

As stated earlier, the optimization procedures implemented in simulation software are all based on metaheuristics and predominantly evolutionary algorithms, which iterate on a *family* of solutions instead of on a single point, and most incorporate some form of memory. The stochastic algorithms, on the other hand, have generally mimicked their deterministic counterparts from nonlinear programming. Since they are also search algorithms, there is also the issue of using the current point versus using past solutions. The use of memory is more obvious in the deterministic case: At a minimum, the best solution(s) obtained up to that point should be recorded somewhere (albeit not necessarily included in the current set of candidate solutions). In the stochastic case, it may not necessarily be beneficial to keep track of such solutions, especially if the performance estimate is very noisy.

The use of long runs or many replications clearly reduces the noise and brings the stochastic setting closer to the deterministic domain. Random search algorithms often count the number of visits to promising configurations.

In addition to the use of a family of points, the algorithms implemented in software are primarily based on discrete search strategies, and, as such, define their own neighborhood structure, not assuming (or exploiting), though possibly inheriting, order inherent in the variable space, e.g., on the real line. Exploiting such order is clearly used in continuous optimization algorithms. The generality gained by the metaheuristic approaches may come at the cost of efficiency for problem settings with structure. For example, discrete-event system models clearly exhibit certain characteristics that may be amenable to more efficient search such as algorithms based on gradient information.

A useful notion defined in statistical ranking and selection procedures is the concept of *correct selection* and the computation, or bounding, of its probability. Correct selection refers to choosing a configuration of the input variables that optimizes the objective function. Many iterative search algorithms have an *asymptotic* probability of correct selection of 1. Ranking and selection procedures introduce the concept of an *indifference zone*, say ϵ, which provides a measure of closeness that the decision maker tolerates away from absolute optimality. This is analogous to the *good enough* sets defined in ordinal optimization. In this case, correct selection corresponds to either choosing a configuration with the optimum value of the objective function or a configuration whose objective function value is within ϵ of the optimal value. The usual statement of performance guarantees for these procedures is to specify a lower bound on the probability of correct selection, in the form of a $(1-\alpha)100\%$ confidence level.

We revisit the issue of convergence and statistical validity in more detail now, in preparation for the description of these procedures in the next section. In words, the algorithms in the research literature provide the following:

• Stochastic Optimization Procedures (e.g., stochastic approximation, random search, sample path optimization): convergence to a true optimum (but possibly only local) under some probabilistic metric.

• Ranking and Selection Procedures: selection of a best solution (or set of best solutions) at some prespecified statistical level.

Typical examples of research results take the following form:

• Stochastic Optimization Procedures:

$$\theta_n \longrightarrow \theta^* \text{ w.p. } 1,$$

which is also known as almost sure (a.s.) convergence. Other common forms of convergence include convergence in probability (measure) and convergence in distribution (also known as weak convergence, a term that confusingly enough does not correspond to the weak law of large numbers, which instead is a convergence-in-probability result). Defining these rigorously is beyond the scope of this article (refer to Wolff 1989, Sec.1–16, for example). In the next section, we review some of the modes via some well-known examples.

• Ranking and Selection Procedures: Probability that the selected θ is within ϵ of the best is at least $(1-\alpha)$.

In contrast, very little in the way of theoretical convergence results exists for the metaheuristics (refer to Table 2) in the deterministic framework; none that the author is aware of in the stochastic environment.

2.2. A Brief Tutorial

This section reviews rudimentary material on probabilistic convergence modes and simulation output analysis at the minimum level required for reading this article.

2.2.1. A Very Basic Primer on Simulation Output Analysis. Simulationists can (and should) skip this subsection, which diverges to speak to those with little simulation background or those on the deterministic side who did not listen very carefully to their simulation professor. The reader is referred to Law and Kelton (2000) or Banks et al. (2000) for further in-depth coverage.

The message is this basic tenet of statistical output analysis:

> Simulation estimates should be accompanied with some indication of precision!

The usual textbook instruction is to provide confidence intervals, e.g., a 95% confidence interval for mean response time is 97 seconds ± 5 seconds. A less strict, implicit means of providing a rough indication of precision is the reporting of significant digits. Under this convention, if the single number 97 seconds is presented (e.g., to upper-level management, who would just as soon not see the ± 5 seconds), the presumption should be that the precision is somewhere on the order of 1 to 10 seconds. In other words, do not report the number 97.395 seconds alone (97.395 seconds ± 5.321 seconds is acceptable, but not as preferred as either 97 ± 5 or 97 ± 5.3) unless the precision extends to the third decimal place, because it is extremely misleading! Unfortunately, the software makes reporting large number of decimal places too easy for the user.

For estimating a single performance measure, a good rough measure of precision is provided by the standard error

$$s/\sqrt{n},$$

where s is the sample standard deviation and n is the number of simulation replications. An approximate 95% confidence interval is constructed by taking two standard errors on both sides of the sample mean. Such an interval is not appropriate when the estimated performance is heavily skewed (e.g., for rare-event measures).

For comparing performances of various designs, one uses pairwise comparisons. Individual measures of precision on each pair can be found and combined with the Bonferroni inequality to get an overall confidence level lower bound, or a simultaneous confidence level can be obtained that coordinates all of the pairs.

Consider just two systems. The most easily applied method to compare them is to use the paired-t confidence interval to check the direction of the sign (technically, checking the hypothesis of whether or not the difference in the means is zero). If the confidence interval contains zero, then, statistically speaking, there is no difference. In fact, it is almost always the case in practice that the analyst believes that there is a difference, and it is the direction that is to be inferred. In order to make this inference, the analyst desires a confidence interval that doesn't contain zero. To achieve this clearly depends on the following factors:

- actual difference in the means;
- variance of each of the estimators;
- covariance between the estimators.

If an efficient *direct* estimator for the difference could be found, that would be ideal. In practice, the analyst takes the individual estimators and forms an estimator for the difference by taking the difference between the two in the obvious manner. Inducing positive correlation will reduce the size of the confidence interval, since

$$\begin{aligned} Var(X-Y) &= Var\,X + Var\,Y - 2\,Cov(X,Y) \\ &< Var\,X + Var\,Y \quad \text{if} \quad Cov(X,Y) > 0. \end{aligned}$$

This is the main idea behind the method of common random numbers, but it is also the basis for other schemes to effectively couple the individual underlying stochastic processes.

2.2.2. Review of Convergence Modes. By way of three well-known examples in classical statistics and one less widely known result, we compare various forms of convergence that are found in research results on stochastic optimization. Let $\overline{X}_n$ denote the sample mean over n i.i.d. samples $\{X_i\}$ with common mean μ and variance σ^2, with $N(\mu, \sigma^2)$ denoting the normal distribution with mean μ and variance σ^2.

- Strong Law of Large Numbers (SLLN)

$$\overline{X}_n \longrightarrow \mu \text{ w.p. 1.}$$

- Weak Law of Large Numbers (WLLN)

$$\overline{X}_n \longrightarrow \mu \text{ in probability.}$$

- Central Limit Theorem (CLT)

$$\overline{X}_n \longrightarrow N(\mu, \sigma^2/n) \text{ in distribution.}$$

- Large Deviations Type of Result

$$P(\overline{X}_n \notin (\mu-\epsilon, \mu+\epsilon)) \longrightarrow e^{-nf(\epsilon)},$$

e.g., if $X_i \sim N(\mu, \sigma^2)$, then $f(x) = (x/\sigma)^2/2$.

All of these results make statements about the convergence of the sample mean to the true mean for a large enough number of samples. Although the CLT has a weaker type of convergence than the two laws of large numbers, it is a stronger result, because it provides the actual asymptotic distribution, while still guaranteeing convergence to the mean, since the variance σ^2/n goes to 0 as n goes to infinity. This variance term in fact provides an estimate of the precision or distance of the sample mean from the true mean for a large number of samples. In other words, it provides the well-known $O(1/\sqrt{n})$ convergence rate for Monte Carlo simulation.

The last result makes a probability statement on the deviation of the sample mean from the true mean. As the number of samples gets larger, the probability of a large deviation from the true mean vanishes exponentially fast. Note that although this also provides a convergence rate, it differs from the CLT result in that the rate is with respect to a probability rather than for estimation per se. In a very crude sense, one can view this as refining the WLLN result with a convergence rate, whereas the CLT result provides a convergence rate for the SLLN version.

Picking the **arg min** can be associated with the last result, whereas actually estimating the optimal value has a convergence rate governed by the CLT result. Hence, finding the optimum may result in exponential convergence, whereas estimating the corresponding value is constrained by the standard canonical inverse square root rate of Monte Carlo simulation referred to above. Clearly, this distinction is absent in deterministic optimization, but the currently implemented optimization routines *completely ignore this concept!*

3. Simulation Optimization Research

The most relevant topics in the research literature include the following:

- ranking and selection, multiple comparison procedures, and ordinal optimization;
- stochastic approximation (gradient-based approaches);
- (sequential) response surface methodology;
- random search;
- sample path optimization (also known as stochastic counterpart).

This section includes a summary of the main ideas and a brief survey of some of the research results.

3.1. Ranking and Selection and Ordinal Optimization

We begin with relevant work that focuses on the comparison theme rather than the search algorithms, because this is a central issue in optimization for simulation that practice has not fully addressed (or exploited). Although usually listed separately from simulation optimization in simulation textbooks or handbook chapters (e.g., Law and Kelton 2000, Banks et al. 2000, Banks 1998), ideas from the ranking and selection (R&S) literature (taken here to include multiple comparison procedures), which uses statistical analysis to determine ordering, have important implications for optimization. The primary feature differentiating R&S procedures (see, for example, Goldsman and Nelson 1998) from optimization procedures is that the R&S procedures evaluate exhaustively all members from a *given* (fixed and finite) set of alternatives, whereas optimization procedures attempt to *search* efficiently through the given set (possibly implicitly defined by constraints) to find improving solutions, because exhaustive search is impractical or impossible (e.g., if the set is unbounded or uncountable). As a result, R&S procedures focus on the comparison aspect, which is a statistical problem unique to the stochastic setting. Clearly, statistics (and probability theory) must also come into play if any convergence results are to be rigorously established for the search algorithms. Thus, these procedures should play a major role in optimization for simulation.

Two important concepts in the methodology have to do with user specification of the following levels:

- an indifference zone (level of precision);
- a confidence level (probability of correct selection).

At a certain level, these are analogous to confidence intervals in estimation, except that both have to be specified here, whereas in estimation the analyst specifies either a precision or a confidence level, and the other follows. To illustrate for the single-server

queue example, an estimation goal might be to estimate mean response time within a precision of plus or minus 10 seconds, or with a 95% confidence level, whereas the selection criterion might be to select an operator that results in an average daily operating cost within \$10 of optimality at a 95% confidence level.

The key idea behind the so-called ordinal optimization approach (Ho et al. 1992, 2000) is that it is much easier to estimate approximate relative order than precise (absolute) value. In addition, there is *goal softening*; instead of looking for the best, one settles for a solution that is *good enough*, which would have to be statistically defined. The former has a common goal with multiple comparison procedures, while the latter is clearly similar in spirit to the indifference zone approach of R&S procedures.

3.1.1. Exponential Convergence of Ordinal Comparisons. The term *output analysis* in stochastic simulation refers to the statistical analysis of simulation output over a set of simulation replications (samples) or one long simulation, with the most basic result being the $O(1/\sqrt{n})$ convergence rate of Monte Carlo estimation. However, *comparisons* often exhibit asymptotic convergence rates that are exponential.

Example: Two Configurations. Consider the simple "optimization" problem of determining which of two configurations has the smallest mean of the output performance measure, where only samples are available to estimate the mean. For simplicity, the performance measures are taken to be normally distributed (unbeknownst to the optimizer analyst):

$$\Theta = \{\theta_a, \theta_b\}, \quad L(\theta_a) \sim N(0, 1/4),$$
$$L(\theta_b) \sim N(d, 1/4), \qquad d > 0.$$

In this case, the optimal configuration is θ_a, since $J(\theta_a) = E[L(\theta_a)] = 0 < d = E[L(\theta_b)] = J(\theta_b)$. With just two configurations, the optimization problem can be reduced to simply determining whether the difference of the means is positive or negative. To this end, define the difference random variable:

$$X = L(\theta_b) - L(\theta_a).$$

Assuming that n independent pairs of samples are taken from each configuration, let X_i denote the ith such sample, following distribution $N(d, 1/2)$. Let ϵ denote the indifference amount. Then one approach would be to estimate each of the two configurations independently until two standard errors for each estimate is less than ϵ. After that, the two means are compared to decide which is smaller.

For illustrative purposes, consider a numerical example where $\epsilon = 0.1$. Then it would take approximately 100 samples to achieve the desired precision for each configuration, regardless of the value of d. What about the probability of correct selection? We have said that the complement of this probability decreases to zero exponentially. However, the *rate* of this decay depends on the value of d. For this simple example, these probabilities are easily calculated using the standard normal cumulative distribution values. Figure 5 illustrates the convergence rate of this probability of correct selection for difference values of d (0.01, 0.1, 0.2, 0.3, 1.0), with the $1/\sqrt{n}$ convergence rate also included for comparison. Thus, to achieve roughly comparable 95% probability of correct selection would require approximately 34 simulations for $d = 0.2$, 15 simulations for $d = 0.3$, and only 2 simulations for $d = 1.0$, significantly less than the 100 simulations required in the naïve approach. Note that when

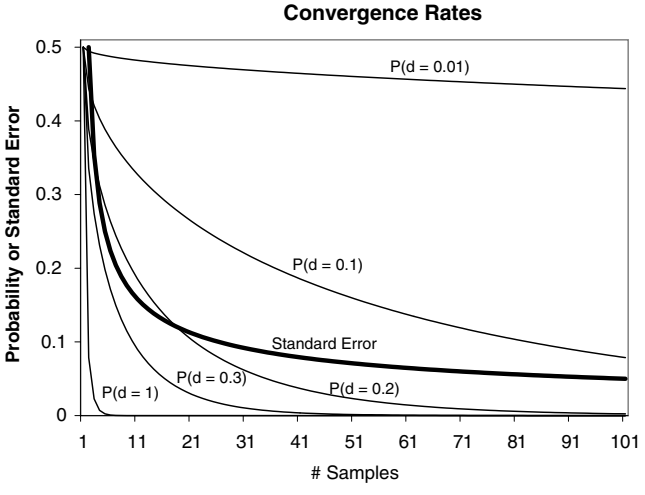

Figure 5 Convergence Rates: Probability of Correct Selection Versus Estimation

comparing the estimation accuracy and the probabilities using the graph, consider only the rates, not actual values, as the values in the graph were chosen so that they could be placed on the same graph with the same scale. The graph shows that when $d > \epsilon$, the advantage of the exponential convergence rate is clear, but decreases when $d \approx \epsilon$. When $d \ll \epsilon$, the exponential decay parameter can be quite close to zero, so that the convergence rate looks more linear than exponential ($e^{-kx} \approx 1 - kx$); however, in this domain there is less concern if the wrong decision is made.

Intuitively, comparison is generally easier than estimation. Think of deciding between two bags of gold sitting before you. You are allowed to lift each of them separately or together. Clearly, it is far easier to determine which is heavier than to determine an actual weight. If the two bags are very close in weight, then it doesn't matter so much if you pick the wrong one (unless you are extremely greedy!).

3.1.2. Variance Reduction Techniques Can Make a Difference! In the simulation community, it is well known that variance reduction techniques such as common random numbers can substantially reduce computational effort. This should be exploited in optimization as well. Here is a dramatic illustration, using the two configurations example again, but this time with the underlying distributions exponentially distributed:

$$L(\theta_a) \sim \exp(\theta_a), \quad L(\theta_b) \sim \exp(\theta_b), \qquad \theta_a < \theta_b,$$

where $\exp(\theta)$ denotes an exponential distribution with mean θ, so again the optimal (minimum-mean) configuration is θ_a. This time, instead of independently generated samples as before, assume that the samples are generated in pairs using common random numbers in the natural way:

$$L(\theta_a, \omega_i) = -\theta_a \ln U_i,$$
$$L(\theta_b, \omega_i) = -\theta_b \ln U_i, \quad U_i \sim U(0, 1),$$

where $U(0, 1)$ denotes a random number uniformly distributed on the interval [0, 1]. Then it is clear that due to the monotonicity properties of the transformation, the probability of correct selection is 1, i.e.,

$$P(L(\theta_a, \omega_i) < L(\theta_b, \omega_i)) = 1,$$

so that coupling through common random numbers has basically reduced the variance to zero as far as the comparison goes!

3.2. Stochastic Approximation

The method of stochastic approximation (SA) dates back over half a century. The algorithm attempts to mimic the gradient search method in deterministic optimization, but in a rigorous statistical manner taking into consideration the stochastic setting. The general form of the algorithm takes the following iterative form (sign would be changed for a maximization problem):

$$\theta_{n+1} = \Pi_\Theta(\theta_n - a_n \hat{\nabla} J(\theta_n)), \tag{3}$$

where Π_Θ denotes some projection back into the constraint set when the iteration leads to a point outside the set (e.g., the simplest projection would be to return to the previous point), a_n is a step size multiplier, and $\hat{\nabla} J$ is an estimate for the gradient of the objective function with respect to the decision variables. In the case of the toy single-server queue example, the iteration would proceed as follows:

$$\theta_{n+1} = \Pi_\Theta\big(\theta_n - a_n\big[\widehat{W}'(\theta_n) - c/\theta^2\big]\big), \tag{4}$$

with the need to find an appropriate $\widehat{W}'$.

For the (s, S) inventory example, Figures 6 and 8 illustrate the typical progression of iterates for a stochastic approximation and sequential response surface methodology procedure (to be discussed in the

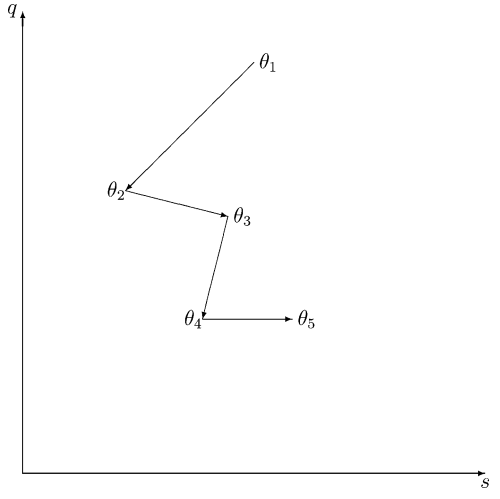

Figure 6 Illustration of a Stochastic Approximation Algorithm

next section), respectively, where $\theta = (s, q)$ and $q = S - s$ is the second parameter optimized in place of S, so that the constraint region is simply the first quadrant (to enforce the constraint $S \geq s$). Note that the two figures imply a continuous optimization problem, but in fact this problem is often posed in a discrete setting (e.g., s and S are integral amounts), as it appeared in the research literature in its original stochastic dynamic programming formulation.

Because of its analogy to steepest descent gradient search, SA is geared towards continuous variable problems, although there has been work recently applying it to discrete variable problems (e.g., Gerencsér 1999). Under appropriate conditions, one can guarantee convergence to the actual minimum with probability one, as the number of iterations goes to infinity. Because of the estimation noise associated with stochastic optimization, the step size must eventually decrease to zero in order to obtain convergence w.p. 1 (i.e., $a_n \longrightarrow 0$), but it must not do so to rapidly so as to converge prematurely to an incorrect point (e.g., $\sum_n a_n = \infty$ is a typical condition imposed, satisfied by the harmonic series $a_n = 1/n$). In practice, the performance of the SA algorithm is quite sensitive to this sequence. Figure 7 illustrates this sensitivity for the (s, S) example, in the case with $a_n = a/n$, where the convergence is highly dependent on the choice of a. Taking the step size constant results at best in weak convergence theoretically (i.e., convergence in distribution, which means that the iterate oscillates or hovers around the optimum), but in practice, a constant step size often results in much quicker convergence in the early stages of the algorithm over decreasing the step size at each step. Robust SA uses the same iterative scheme but returns the *average* of some number of iterates (e.g., moving window or exponentially weighted moving average) as the estimate of the optimum configuration. The averaging serves to reduce the noise in the estimation, leading to a more robust procedure. Again, because it is a gradient search method, SA generally finds local extrema, so that enhancements are required for finding the global optimum.

The effectiveness of stochastic approximation algorithms is dramatically enhanced with the availability of direct gradients, one motivating force behind the

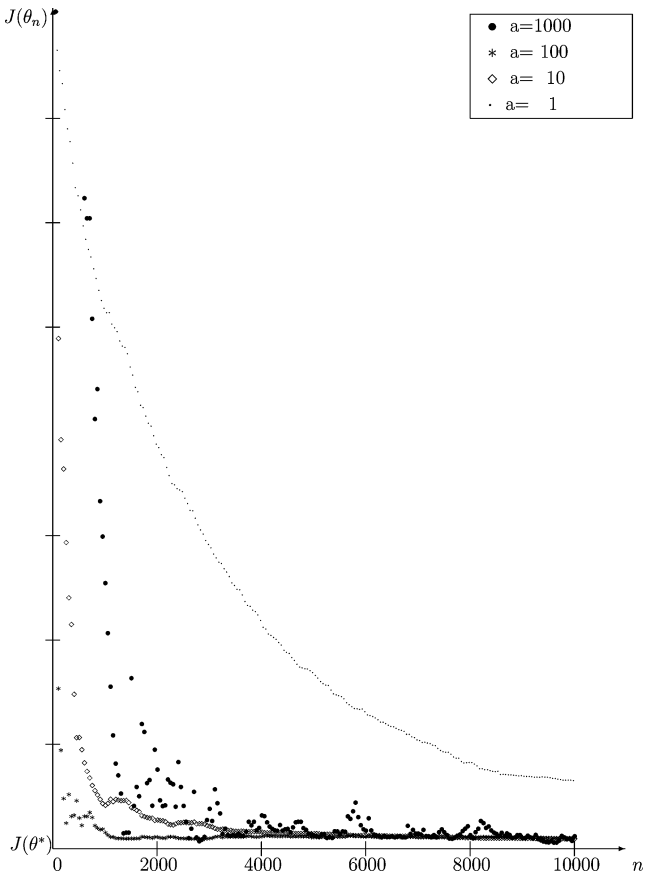

Figure 7 **Effect of Choice of Initial Step Size a (Parameter Updates Every 50 Periods)**

flurry of research in gradient estimation techniques in the 1990s (see the books by Fu and Hu 1997, Glasserman 1991, Ho and Cao 1991, Rubinstein and Shapiro 1993, Pflug 1996). The best-known gradient estimation techniques are perturbation analysis (PA) and the likelihood ratio/score function (LR/SF) method. An example of applying PA and SA to an option pricing problem is given in Fu and Hu (1995). Infinitesimal perturbation analysis (IPA) has been successfully applied to a number of real-world supply chain management problems, using models and computational methods reported in Kapuscinski and Tayur (1999).

If no direct gradient is available, naïve one-sided finite difference (FD) estimation would require $p+1$ simulations of the performance measure (where p is the dimension of the vector θ) in order to obtain a single gradient estimate, i.e., the ith component of the

Table 3 Gradient Estimation Approaches for Stochastic Approximation

Approach	Number of Simulations	Key Features	Disadvantages
IPA	1	Highly efficient, easy to implement	Limited applicability
Other PA	Usually >1	Model-specific implementations	Difficult to apply
LR/SF	1	Requires only model input distributions	Possibly high variance
SD	$2p$	Widely applicable, model-free	Generally noisier
FD	$p+1$	Widely applicable, model-free	Generally noisier
SP	2	Widely applicable, model-free	Generally noisier

gradient estimate based on estimates $\widehat{J}$ of the objective function would be given by

$$(\hat{\nabla} J(\theta))_i = \frac{\widehat{J}(\theta + c_i e_i) - \widehat{J}(\theta)}{c_i},$$

and two-sided symmetric difference (SD) estimation would require $2p$ simulations:

$$(\hat{\nabla} J(\theta))_i = \frac{\widehat{J}(\theta + c_i e_i) - \widehat{J}(\theta - c_i e_i)}{2c_i},$$

where e_i denotes the unit vector in the ith direction. Choice of the difference parameters $\{c_i\}$ must balance between too much noise (small values) and too much bias (large values). In either case, however, the estimate requires $O(p)$ simulation replications. The method of simultaneous perturbations (SP) stochastic approximation (SPSA) avoids this by perturbing in all directions *simultaneously*, as follows:

$$(\hat{\nabla} J(\theta))_i = \frac{\widehat{J}(\theta + \Delta) - \widehat{J}(\theta - \Delta)}{2\Delta_i},$$

where $\Delta = [\Delta_1 \ldots \Delta_p]$ represents a vector of i.i.d. *random* perturbations satisfying certain conditions. The simplest and most commonly used perturbation distribution in practice is the symmetric (scaled) Bernoulli distribution, e.g., $\pm c_i$ w.p. 0.5. Spall (1992) shows in fact that the asymptotic convergence rate using this gradient estimate in a SA algorithm is the same as the naïve method above. The difference in simulations between the FD/SD estimators and the SP estimators is that the numerator, which involves the expensive simulation replications, varies in the FD/SD estimates, whereas the numerator is constant in the SP estimates, and it is the denominator involving the random perturbations that varies. Table 3 provides a brief summary of the main approaches in estimating the gradient for stochastic approximation algorithms, where IPA stands for infinitesimal perturbation analysis. Other methods not listed in the table include frequency domain experimentation and weak derivatives (Pflug 1996).

3.3. Response Surface Methodology

The goal of response surface methodology (RSM) is to obtain an approximate functional relationship between the input variables and the output (response) objective function. When this is done on the entire (global) domain of interest, the result is often called a metamodel (e.g., Barton 1998). This metamodel can be obtained in various ways, two of the most common being regression and neural networks. Once a metamodel is obtained, in principle, appropriate deterministic optimization procedures can be applied to obtain an estimate of the optimum. However, in general, optimization is usually not the primary purpose for constructing a metamodel and, in practice, when optimization is the focus, some form of sequential RSM is used (Kleijnen 1998). A more localized response surface is obtained, which is then used to determine a search strategy (e.g., move in an estimated gradient direction). Again, regression and neural networks are the two most common approaches. Sequential RSM using regression is one of the most established forms of simulation optimization found in the research literature, but it is not implemented in any of the commercial packages. SIMUL8's OPTIMIZ proceeds using a form of sequential RSM using neural networks (`http://www.SIMUL8.com/optimiz1.htm`):

> OPTIMIZ uses SIMUL8's 'trials' facility multiple times to build an understanding of the simulation's 'response surface'. (The effect that the variables, in combination, have on the outcome). It does this very

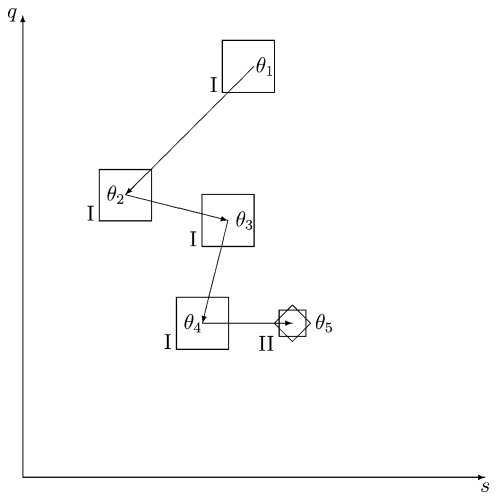

Figure 8 Illustration of a Sequential Response Surface Methodology Procedure

quickly because it does not run every possible combination! It uses Neural Network technology to learn the shape of the response surface from a limited set of simulation runs. It then uses more runs to obtain more accurate information as it approaches potential optimal solutions.

Figure 8 illustrates the sequential RSM procedure using regression for the (s, S) inventory model. The θ_i iterate is in the center of a set of simulated points chosen by design of experiments methodology (i.e., factorial design). In Phase I (the first iterates in the figure), 2^2 points in a square are simulated, and a linear regression is performed to characterize the response surface around the current iterate. A line search is carried out in the direction of steepest descent to determine the next iterate. This process is repeated until the linear fit is deemed inadequate, at which juncture additional points are simulated, and in the single Phase II (fifth iterate in the figure), quadratic regression is carried out to estimate the optimum from the resulting fit.

3.4. Random Search Methods

The advantage of random search methods is their generality and the existence of theoretical convergence proofs. They have been primarily applied to discrete optimization problems recently, although, in principle, they could be applied to continuous optimization problems, as well. A central part of the algorithm is defining an appropriate neighborhood structure, which must be connected in a certain precise mathematical sense. Random search algorithms move iteratively from a current single design point to another design point in the neighborhood of the current point. Differences in algorithms manifest themselves in two main fashions: (a) how the next point is chosen and (b) what the estimate is for the optimal design. For (b), the choice is usually between taking the current design point versus choosing the one that has been visited the most often. The latter is the natural discrete analog to the robust SA approach discussed earlier in the continuous variable setting, where iterates are averaged. Averaging often wouldn't make sense in many discrete settings, where there is no meaningful ordering on the input variables. Conversely, counting wouldn't make sense in the continuous variable setting, where the probability of any particular value is usually zero.

Let $N(\theta)$ denote the neighborhood set of $\theta \in \Theta$. One version of random search that gives the general flavor is the following:

(0) Initialize:
Select initial configuration $\hat{\theta}_*$;
Set $n_{\hat{\theta}_*} = 1$ and $n_\theta = 0 \ \forall \theta \neq \hat{\theta}_*$.

(1) Iterate:
Select another $\theta_i \in N(\hat{\theta}_*)$ according to some prespecified probability distribution.
Perform simulations to obtain estimates $\widehat{J}(\hat{\theta}_*)$ and $\widehat{J}(\theta_i)$.
Increase counter for point with best estimate and update current point: (**1** denotes the indicator function)

$$n_{\hat{\theta}_*} = n_{\hat{\theta}_*} + \mathbf{1}\{\widehat{J}(\hat{\theta}_*) \leq \widehat{J}(\theta_i)\};$$
$$n_{\theta_i} = n_{\theta_i} + \mathbf{1}\{\widehat{J}(\hat{\theta}_*) > \widehat{J}(\theta_i)\};$$
$$\text{If } \widehat{J}(\hat{\theta}_*) > \widehat{J}(\theta_i), \text{ then } \hat{\theta}_* \leftarrow \theta_i.$$

(3) Final Answer:
When stopping rule satisfied, return

$$\theta^* = \arg\max_{\theta \in \Theta} n_\theta.$$

A simple version of this algorithm (Andradóttir 1996) that is guaranteed to converge globally w.p. 1 requires the feasible set Θ to be finite (though possibly large)

and takes the neighborhood of a point θ to be the rest of the feasible set $\Theta \setminus \{\theta\}$, which is uniformly sampled (i.e., each point has an equal probability of being selected). However, even this simple algorithm may face implementation difficulties, as it may not be so easy to sample randomly from the neighborhood with the appropriate distribution (see Banks et al. 2000, p. 495).

3.5. Sample Path Optimization

Sample path optimization (SPO) is a method applicable to (1) that attempts to exploit the powerful machinery of existing deterministic optimization methods for continuous variable problems (e.g., see Gürkan et al. 1999). The framework is as follows: Think of $\omega = (\omega_1, \omega_2, \ldots, \omega_n, \ldots)$ as the set of all possible sample paths for $L(\theta, \omega_i)$. Define the sample mean over the first n sample paths:

$$\overline{L}_n(\theta) = \frac{1}{n} L(\theta, \omega_i).$$

If each of the $L(\theta, \omega_i)$ are i.i.d. unbiased estimates of $J(\theta)$, then by the strong law of large numbers, we have that with probability one,

$$\overline{L}_n(\theta) \longrightarrow J(\theta).$$

SPO simply optimizes, for a sufficiently large n, the deterministic function $\overline{L}_n$, which approximates $J(\theta)$. In the simulation context, the method of common random numbers is used to provide the same sample paths for $\overline{L}_n(\theta)$ over different values of θ. Again, the availability of derivatives greatly enhances the effectiveness of the SPO approach, as many nonlinear optimization packages require these. The chief advantage of SPO is, as Robinson (1996) states, "we can bring to bear the large and powerful array of deterministic optimization methods that have been developed in the last half-century. In particular, we can deal with problems in which the parameters θ might be subject to complicated constraints, and therefore in which gradient-step methods like stochastic approximation may have difficulty." The stochastic counterpart method (Rubinstein and Shapiro 1993) can be viewed as a variant of SPO that explicitly invokes the likelihood ratio method (and importance sampling) to carry out the optimization.

4. Optimization for Simulation Software

This section provides further descriptions (algorithmic details being proprietary) for two of the most popular optimization routines currently available in commercial simulation software (refer to Table 1). The description of AutoStat is based on Bitron (2000). The description of OptQuest is based on Glover et al. (1999).

• **AutoStat**—This is a statistical analysis package available with AutoMod (and its more specialized version AutoSched), a simulation software environment provided by AutoSimulations, Inc., a company that has perhaps the largest market share in the semiconductor manufacturing industry. The optimization routine, which is just one part of the AutoStat suite of statistical output analysis tools (other features include design of experiments, warm-up determination, confidence intervals, and factor-response analysis), incorporates an evolutionary strategies algorithm (genetic algorithm variation) and handles multiple objectives by requiring weights to form a fitness function. Design of experiments terminology is used in the dialog boxes (i.e., factors and responses). The user selects the input variables (factors) to optimize and the performances measures (responses) of interest. For each input variable, the user specifies a range or set of values. For each performance measure, the user specifies its relative importance (with respect to other performance measures) and a minimization or maximization goal. The user also specifies the number of simulation replications to use for each iteration in the search algorithm. Further options include specifying the maximum number of total replications per configuration, the number of parents in each generation, and the stopping criteria, which is of two forms: termination after a maximum number of generations or when a specified number of generations results in less than a specified threshold level of percentage improvement. The total number of children is set at seven times the number of parents per generation, the latter of which is also user specified. While the optimization is in progress, the software displays a graph of the objective function value for four measures as a function of the generation number: overall best, best in current generation, parents' average, and children's

average. When complete, the top 30 configurations are displayed, along with various summary statistics from the simulation replications.

• **OptQuest**—This package is a stand-alone optimization routine that can be bundled with a number of the commercial simulation languages, such as the widely used discrete-event simulation environment Arena and the Monte Carlo spreadsheet add-in Crystal Ball. The algorithm incorporates a combination of strategies based on scatter search and tabu search, along with neural networks for screening out candidates likely to be poor. Being a completely separate software package, the algorithm treats the simulation model essentially as a black box, where the focus of the algorithm is on the search and not on the statistics and efficiency of comparison (http://www.opttek.com/optquest/oqpromo.html, November 2000):

> The critical 'missing component' is to disclose which decision scenarios are the ones that should be investigated—and still more completely, to identify good scenarios automatically by a search process designed to find the best set of decisions.

Scatter search is very similar to genetic algorithms, in that both are population-based procedures. However, Glover et al. (1999) claim that whereas naïve GA approaches produce offspring through random combination of components of the parents, scatter search produces offspring more intelligently by incorporating history (i.e., past evaluations). In other words, diversity is preserved, but natural selection is used in reproduction prior to being evaluated. This is clearly more important in the simulation setting, where estimation costs are so much higher than search costs. The makers of OptQuest claim that "it is possible to include any set of conditions that can be represented by a mixed integer programming formulation" (Glover et al. 1999, p. 259). The neural network is basically a metamodel representation, which is used as a screening device to discard points where the objective function value is predicted to be poor by the neural network model, without actually performing any additional simulation. It differs from factor screening in that it screens out individual points, not an entire dimension of the parameter vector. Since the neural network is clearly a rough approximation, both in approximating the objective function and in the uncertainty associated with the simulation outputs, OptQuest incorporates a notion of a *risk* metric, defined in terms of standard deviations. If the neural network predicts an objective function value for the candidate solution that is worse than the best solution up to that point by an amount exceeding the risk level, then the candidate solution is discarded without performing any simulations. This type of intelligent screening is certainly highly desirable. However, its effectiveness was not fully tested in the comparisons with the GA algorithm reported in Glover et al. (1999), because deterministic problems were used. Thus, the discarding is a function only of the goodness of the neural network in approximating the objective function and not of any stochastic behavior associated with simulation.

The focus on search is common among all the commercial packages, again reflecting the optimization-for-simulation practice view of Figure 3, when computation time follows the proportions of Figure 1.

5. Conclusions and Predictions

The current commercial software is a good start, but fails to exploit the research in simulation optimization, from which there are many useful results that have potential to dramatically improve the efficiency of the procedures. Mainly, heuristics from combinatorial (discrete) optimization have been employed, and the effectiveness of these implementations is based primarily on the robustness of the resulting procedures to the noise levels inherent in the stochastic nature of the systems. Working with families of solutions instead of a single point is a primary means by which such robustness is achieved and, in that sense, is closely related to the idea of a good-enough set from ordinal optimization. However, other ideas concerning the faster rate of convergence of ordinal comparisons versus cardinal estimation have yet to be incorporated, which could lead to much more efficient use of computational resources. In other words, the biggest problem with currently implemented methods is that though they may be intelligent in performing the search procedures, they are somewhat oblivious to the stochastic nature of the underlying system. Thus, they completely lack any sense of

how to allocate efficiently a simulation budget. *Precision* of the estimated output performance measure(s) (and especially relative order, as opposed to absolute value) should be used dynamically (as opposed to the current pre-defined static approach), in conjunction with the mean estimates themselves to guide the search and simulation budget allocation simultaneously. Variance reduction techniques should be fruitfully integrated into the simulation-optimization interface, as part of the needs indicated in Figure 4. Lastly, it is a little baffling that sequential RSM using regression—very well established in the literature *and* quite general and easy to implement—has not been incorporated into any of the commercial packages.

On the other hand, much of existing research has concentrated on relatively narrow areas or toy problems, the single-server queue being the most obvious example. While this research does lead to insights, interesting algorithms, and important theoretical convergence results, the work lacks the jump to the next step of practice. Of course, one could argue that this is not the primary goal of research, but this leaves the gap in the middle for the commercial developer as to how to make the apparently nontrivial leap from a single-server queue to a complicated call center. Furthermore, the research results seem to suffer from two extremes: 1) algorithms that work extremely well are too specialized to be practical, or 2) algorithms that apply very generally often converge too slowly in practice. In addition, although the trend has changed a bit in the last few years, historically there has been a much higher concentration of research effort spent on the continuous variable case, when many of the problems that arise in the discrete-event simulation context are dominated by discrete-valued variables.

Here is this author's view on desirable features in a good implementation of optimization for commercial simulation software:

- **Generality.** The optimization routines must be able to handle the wide range of problems that a user is likely to encounter or be interested in applying. This means, for example, that gradient-based algorithms (whether SA or SPO) requiring an unbiased direct gradient estimate have found difficulty in commercial implementation, because they can be very problem specific and hence not easily included in general purpose simulation software. On the other hand, this does not mean such approaches have no place in commercial software either. An analogy in mathematical programming is that of the transportation algorithm; the software should be intelligent enough to be able to check for special structure and exploit it when available. This of course is a non-trivial problem. In a queueing system, e.g., a call center, this might be as simple as being the special case when there is just a single class of customers under FCFS and just one skill level of operators available. It simply means that the user should not need to make this decision, which is part of the point of the next bullet. Furthermore, optimization techniques such as SPSA (Spall 1992, Fu and Hill 1997, Gerencsér 1999), which are not at all model dependent and easy to implement (though there is model tuning, analogous to that in neural networks), seem ripe for commercial adaptation.
- **Transparency to user.** With graphical user interfaces (GUIs) and pull-down menus, the mathematical (without even mentioning statistical) sophistication of users has seen a marked shift downwards. While this has the clear benefit of allowing the power of simulation to reach a much wider audience of users, it also means that any complications associated with optimization must be shielded from the interface.
- **High dimensionality.** It is not clear how well the currently implemented algorithms would perform in higher dimensions, in terms of computational efficiency. Their lack of emphasis on the stochastic nature of the underlying system would be accentuated in this setting. More efficient algorithms that are geared to higher dimensions such as SPSA are definitely worth further investigation.
- **Efficiency.** Moore's law and the resulting advances in computational power being what they have been, the fact remains that many real-world problems are still combinatorial, so that providing a truly integrated simulation optimization routine can lead to more efficient use of computational resources, resulting in good solutions for larger problems.

Currently available software does a good job on the first two items, and it is probably for those reasons that they have enjoyed relative success. The last two items, although important, do not hit the user as

directly in the beginning. To better bridge theory and practice, the author believes the following challenges need to be addressed (again refer to Figure 4):

- providing some measure of goodness (other than just improvement over the starting point, which most packages provide) for the metaheuristics that dominate the commercial field;
- developing practical and effective implementation of algorithms with proven convergence properties that dominate the research literature.

As stated already, an obvious linkage can come from R&S procedures and the related ideas of ordinal optimization, where the aim of the former is to provide statistical guarantees of optimal selection and the latter aims to do this efficiently, albeit sometimes heuristically. In other words, by treating the simulation model in the way that the metaheuristic approaches are generally applied, there is an immense waste of simulation replications used to obtain precise estimates at variable settings whose poor relative performance becomes apparent with just a few replications. The commercial package OptQuest attempts to compensate for this by using a neural network metamodel to screen out such candidates.

In some sense, ordinal optimization is an amalgamation of a number of disparate good ideas for stochastic optimization. The notion of a good-enough set has parallels with the family of solutions retained in deterministic-based evolutionary algorithms such as genetic algorithms. Concentrating on ordinal comparisons rather than cardinal estimation is certainly related to the rigorous statistical procedures developed in the R&S literature. The optimal budget computing allocation (OBCA) approach (Chen, Chen, and Yucesan 2000, Chen et al. 2000) is one link between these two, and would seem to be a candidate for commercial development due to its relative ease of implementation.

Related to the philosophy of avoiding wasted simulation computations is the idea of factor screening (factor analysis), another well-established domain in design of experiments methodology. The main idea of factor screening is to identify which input variables have the largest effect on the output response (objective function). This can help to reduce dimensionality. Another useful approach with roots in experimental design, very relevant to response surface methodology, is robust design (see Sanchez 2000).

Parallel computing is another avenue that has yet to be fully exploited, although many of the methods, notably ordinal optimization and design of experiments (with multiple comparisons), clearly lend themselves to taking advantage of this paradigm.

Although theoretical asymptotic convergence results are elegant in terms of research, a practical difficulty that arises in implementing search algorithms, whether they are based on metaheuristics, stochastic approximation, or random search, is deciding when to stop, i.e., choosing a stopping rule. This is less of an issue in deterministic optimization, e.g., gradient-based algorithms can stop when the gradient is zero. In the stochastic setting, zero gradient could be due to noise, not truly indicative of having reached a (local) extremum. Intuitively, the stopping-rule problem would be addressed by defining some appropriate measure that determines when further iteration seems futile, but often in software it is handled by simply specifying the number of iterations or perhaps the total number of simulation replications; in other words specifying some sort of computer budget. How to do this most efficiently is clearly important, and is not well addressed in existing software, but clearly it is closely related to work in optimal computing budget allocation mentioned earlier.

On the research side, there is little in the way of good algorithms to handle random constraints, i.e., those cases where the constraint set and not just the objective function involves quantities that must be estimated. As discussed earlier, the (s, S) inventory example is often formulated this way. Optimization problems involving queueing systems (e.g., the call center or a manufacturing system) are often of the form of maximizing throughput (number of calls handled, number of parts produced) subject to a constraint on the probability that a customer (part) will have to wait more than a certain amount of time in queue (or a lead time constraint for finished goods). Both throughput and waiting time are performance measures to be estimated from the simulation.

One item of note that came out of the panel discussion at the 2000 Winter Simulation Conference (Fu et al. 2000) was the desire (or need?) to have

a standard set of problems (testbed) on which to compare various algorithms. As all readers of the *INFORMS Journal on Computing* can empathize, this is not unique to the simulation optimization community but common to all computational algorithmic developments in the interface between OR and CS. The caution inherent in establishing such a standardized testbed remains the same: to avoid developing algorithms tuned to the particular set of problems.

Space and scope limitations preclude discussion of some newly developed techniques in combinatorial optimization that may hold potential for application to the stochastic simulation setting, but two fairly recent approaches that have showed promise include ant colony optimization (see Dorigo and Di Caro 1999; Bonabeau et al. 1999; Corne et al. 1999) and nested partitions (Shi and Olafsson 2000).

6. Probing Further

As mentioned earlier, the chapter by Andradóttir (1998) in the *Handbook of Simulation* and the survey article by Fu (1994) in the *Annals of OR* are good places to begin to delve further in depth into the simulation optimization research literature (see also Swisher et al. 2001 for further updated references and also the earlier article by Jacobson and Schruben 1989). The Winter Simulation Conference Proceedings article by Fu et al. (2000) provides position statements from a panel of simulation researchers' and practitioners' diverse set of perspectives.

Online resources includes the Winter Simulation Conference (`http://www.wintersim.org`) and the INFORMS College on Simulation (`http://www.informs-cs.org`), which contains a host of useful links, including all of the Winter Simulation Conference Proceedings articles from 1997 onward (`http://www.informs-cs.org/wscpapers.html`).

The most basic result is the $O(1/\sqrt{n})$ convergence rate of estimation via Monte Carlo simulation. One way to improve upon this rate is quasi-Monte Carlo simulation, an approach that uses quasi-random numbers (see Niederreiter 1992, Niederreiter and Spanier 2000) to generate the underlying randomness rather than pseudo-random numbers. For recent developments on this flourishing area of research that has developed over the past decade or so (though quasi-random numbers themselves have been around longer), refer to the bi-annual Conference on Monte Carlo and quasi-Monte Carlo methods (`http://www.mcqmc.org/`). Exponential convergence rate results stem from large deviations theory; see Dembo and Zeitouni (1998), Shwartz and Weiss (1998), Bucklew (1990), and Varadhan (1984) for books on the general subject and see Dai (1996) and Dai and Chen (1997) for specific application to the simulation context.

Useful books on R&S and multiple comparison procedures are Bechofer et al. (1995) and Hochberg and Tamhane (1987). One recent effort to combine R&S procedures with an efficient search procedure is Scenario Seeker, developed by Justin Boesel (winner of the 1999 Dantzig Dissertation Award for his work) and Barry Nelson (Boesel 1999; Boesel, Nelson, and Ishii 2001; Boesel, Nelson, and Kim 2001; see also Goldsman et al. 1999 and Nelson et al. 2001). This routine, written for the AweSim! simulation environment (Symix Advanced Planning & Scheduling Division, formerly Pritsker Corporation, `http://www.pritsker.com`), also uses a heuristic algorithm for the search, with efficient allocation of simulation replications incorporated into the search phase. Statistical validity for the offered solution are provided using R&S procedures; in particular, initial screening via subset selection to reduce a possibly large set of configurations to a more manageable size, followed by a standard two-stage R&S procedure to select the best. The software is not available commercially and is owned by JGC Corporation, a Japanese construction management and consulting firm.

Acknowledgments

This work was supported in part by the National Science Foundation under Grants DMI-9713720 and DMI-9988867 and by the Air Force Office of Scientific Research under Grant F496200110161. The author thanks the two referees, the Area Editor, and the Feature Article Editor for their comments that have led to an improved exposition.

References

Andradóttir, S. 1996. A global search method for discrete stochastic optimization. *SIAM Journal on Optimization* **6** 513–530.

Andradóttir, S. 1998. Simulation optimization. Chapter 9 in J. Banks, ed. *Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice*. John Wiley & Sons, New York.

Banks, J., ed. 1998. *Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice*. John Wiley & Sons, New York.

Banks, J., J. S. Carson, B. L. Nelson, D. M. Nicol. 2000. *Discrete Event Systems Simulation*, 3rd ed. Prentice Hall, Englewood Cliffs, NJ.

Barton, R. 1998. Simulation metamodels. *Proceedings of the Winter Simulation Conference*. 167–174.

Bechhofer, R. E., T. J. Santner, D. M. Goldsman. 1995. *Design and Analysis of Experiments for Statistical Selection, Screening, and Multiple Comparisons*. John Wiley & Sons, New York.

Bitron, J. February 2000. Optimizing AutoMod Models with AutoStat. *AutoFlash* Monthly Newsletter, AutoSimulations, Inc., Bountiful, UT.

Boesel, J. 1999. *Search and Selection for Large-Scale Stochastic Optimization*. Ph.D. dissertation, Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL.

Boesel, J., B. L. Nelson, N. Ishii. 2001. A framework for simulation-optimization software. *IIE Transactions*, forthcoming.

Boesel, J., B. L. Nelson, S.-H. Kim. 2001. Using ranking and selection to 'clean up' after simulation optimization. submitted for publication, *Operations Research*.

Bonabeau, E., M. Dorigo, T. Theraulaz. 1999. *From Natural to Artificial Swarm Intelligence*. Oxford University Press, New York.

Bucklew, J. A. 1990. *Large Deviations Techniques in Decision, Simulation, and Estimation*. John Wiley & Sons, New York.

Chen, H. C., C. H. Chen, E. Yücesan. 2000. Computing efforts allocation for ordinal optimization and discrete event simulation. *IEEE Transactions on Automatic Control* **45** 960–964.

Chen, H. C., J. Lin, E. Yücesan, S. E. Chick. 2000. Simulation budget allocation for further enhancing the efficiency of ordinal optimization. *Journal of Discrete Event Dynamic Systems: Theory and Applications* **10** 251–270.

Corne, D., M. Dorigo, F. Glover, eds. 1999. *New Ideas in Optimisation*. McGraw-Hill, New York.

Dai, L. 1996. Convergence properties of ordinal comparison in the simulation of discrete event dynamic systems. *Journal of Optimization Theory and Applications* **91** 363–388.

Dai, L., C. Chen, 1997. Rate of convergence for ordinal comparison of dependent simulations in discrete event dynamic systems. *Journal of Optimization Theory and Applications* **94** 29–54.

Dembo, A., O. Zeitouni. 1998. *Large Deviations Techniques and Applications*, 2nd ed. Springer-Verlag, Berlin, Germany.

Dorigo, M., G. Di Caro. 1999. The ant colony optimization meta-heuristic. D. Corne, M. Dorigo, F. Glover, eds. *New Ideas in Optimization*. McGraw-Hill, New York. 11–32.

Fu, M. C., 1994. Optimization via simulation: A review. *Annals of Operations Research* **53** 199–248.

Fu, M. C., S. Andradóttir, J. S. Carson, F. Glover, C. R. Harrell, Y. C. Ho, J. P. Kelly, S. M. Robinson. 2000. Integrating optimization and simulation: research and practice. *Proceedings of the Winter Simulation Conference*. 610-616. http://www.informs-cs.org/wsc00papers/082.PDF.

Fu, M. C., S. D. Hill. 1997. Optimization of discrete event systems via simultaneous perturbation stochastic approximation. *IIE Transactions* **29** 233–243.

Fu, M. C., J. Q. Hu. 1995. Sensitivity analysis for Monte Carlo simulation of option pricing. *Probability in the Engineering and Information Sciences* **9** 417–446.

Fu, M. C., J. Q. Hu. 1997. *Conditional Monte Carlo: Gradient Estimation and Optimization Applications*. Kluwer Academic, Boston, MA.

Gass, S. I., C. M. Harris, eds. 2000. *Encyclopedia of Operations Research and Management Science*, 2nd ed. Kluwer Academic, Boston, MA.

Gerencsér, L. 1999. Optimization over discrete sets via SPSA. *Proceedings of the IEEE Conference on Decision and Control*, 1791–1795.

Glasserman, P. 1991. *Gradient Estimation Via Perturbation Analysis*. Kluwer Academic, Boston, MA.

Goldsman, D., B. L. Nelson. 1998. Comparing systems via simulation, Chapter 8 in J. Banks, ed. *Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice*. John Wiley & Sons, New York, 273–306.

Glover, F., J. P. Kelly, M. Laguna. 1999. New advances for wedding optimization and simulation. *Proceedings of the Winter Simulation Conference*. 255–260.

Goldsman, D., B. L. Nelson, T. Opicka, A. A. B. Pritsker. 1999. A ranking and selection project: Experiences from a university-industry collaboration. *Proceedings of the Winter Simulation Conference*. 83–92.

Gürkan, G., A. Y. Özge, S. M. Robinson. 1999. Sample-path solution of stochastic variational inequalities. *Mathematical Programming* **84** 313–333.

Hochberg, Y., A. C. Tamhane. 1987. *Multiple Comparison Procedures*. John Wiley & Sons, New York.

Ho, Y. C., X. R. Cao. 1991. *Discrete Event Dynamics Systems and Perturbation Analysis*. Kluwer Academic, Boston, MA.

Ho, Y. C., C. G. Cassandras, C. H. Chen, L. Y. Dai. 2000. Ordinal optimization and simulation. *Journal of Operations Research Society* **51** 490–500.

Ho, Y. C., R. Sreenivas, P. Vakili. 1992. Ordinal optimization of DEDS. *Discrete Event Dynamic Systems: Theory and Applications* **2** 61–88.

Jacobson, S. H., L. W. Schruben. 1989. A review of techniques for simulation optimization. *Operations Research Letters* **8** 1–9.

Kapuscinski, R., S. R. Tayur. 1999. Optimal policies and simulation based optimization for capacitated production inventory systems. Chapter 2 in S. R. Tayur, R. Ganeshan, M. J. Magazine, eds. *Quantitative Models for Supply Chain Management*. Kluwer Academic, Boston, MA.

Kleijnen, J. P. C. 1998. Experimental design for sensitivity analysis, optimization, and validation of simulation models. Chapter 6

in J. Banks, ed. *Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice*. John Wiley & Sons, New York.

Law, A. M., W. D. Kelton. 2000. *Simulation Modeling and Analysis*, 3rd ed. McGraw-Hill, New York.

Nelson, B. L., J. Swann, D. Goldsman, W. Song. 2001. Simple procedures for selecting the best simulated system when the number of alternatives is large. *Operations Research* **49** 950–963.

Niederreiter, H. 1992. *Random Number Generation and Quasi-Monte Carlo Methods*. SIAM, Philadelphia, PA.

Niederreiter, H., J. Spanier, eds. 2000. *Monte Carlo and Quasi-Monte Carlo Methods*. Springer, New York.

Pflug, G. C. 1996. *Optimization of Stochastic Models*. Kluwer Academic, Boston, MA.

Robinson, S. M. 1996. Analysis of sample-path optimization. *Mathematics of Operations Research* **21** 513–528.

Rubinstein, R. Y., A. Shapiro. 1993. *Discrete Event Systems: Sensitivity Analysis and Stochastic Optimization by the Score Function Method*. John Wiley & Sons, New York.

Sanchez, S. M. 2000. Robust design: seeking the best of all possible worlds. *Proceedings of the Winter Simulation Conference*. 69–76. http://www.informs-cs.org/wsc00papers/013.PDF.

Shi, L., S. Olafsson. 2000. Nested partitioned method for global optimization. *Operations Research* **48** 390–407.

Shwartz, A., A. Weiss. 1998. *Large Deviations for Performance Analysis: Queue, Communications, and Computing*, 2nd ed. Springer-Verlag, Berlin, Germany.

Spall, J. C. 1992. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE Transactions on Automatic Control* **37** 332–341.

Swisher, J. R., P. D. Hyden, S. H. Jacobson, L. W. Schruben. 2001. Discrete-event simulation optimization: a survey of recent advances. *IIE Transactions*. Submitted.

Varadhan, S. 1984. *Large Deviations and Applications*. SIAM, Philadelphia, PA.

Wolff, R. W. 1989. *Stochastic Modeling and the Theory of Queues*. Prentice Hall, Englewood Cliffs, NJ.

Accepted by Edward A. Wasil; received October 2000; revised May 2001, August 2001; accepted April 2002.