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

SIMULATION OPTIMIZATION

Michael C. Fu

Robert H. Smith School of Business
Van Munching Hall
University of Maryland
College Park, MD 20742-1815, U.S.A.

ABSTRACT

In this tutorial introduction to simulation optimization, we present motivating and illustrative examples, summarize most of the major approaches, and briefly describe some software implementations. The focus is on issues and concepts, rather than mathematical rigor, so the format is Q & A rather than theorem-proof.

1 INTRODUCTION

Q: What is simulation optimization?
Or at least what do we mean by the term in the context of this tutorial?

A: Optimization of performance measures based on outputs from stochastic (primarily discrete-event) simulations (see also Fu 2001a).

Q: What is the difference between stochastic optimization and simulation optimization?

A: These terms are very closely related and sometimes used interchangeably, but as noted above, our focus is on outputs from stochastic discrete-event simulation models, whereas stochastic optimization is generally more broadly defined, encompassing any system involving stochastic behavior (e.g., continuous-time models or actual functioning systems). Furthermore, in the usual context of simulation optimization, simulation is extremely expensive (computationally) relative to optimization; in other words, the computational requirements of a single replication of the simulation model of interest are likely to exceed the typical computation time of any medium-sized (thousands of variables) linear program.

Q: Algorithms such as simulated annealing use randomness, so are they stochastic optimization procedures?

A: One needs to be careful in distinguishing between a method that is specifically designed to attack stochastic problems — which is what is generally defined as a stochastic optimization procedure — and a method that uses stochastic properties in its search, which is what simulated annealing and most of the modern versions of metaheuristics do. These procedures could also be adapted for use for stochastic optimization (and in fact have been, see Table 1 later on), but they were not originally formulated to address problems for which only estimates of the objective function were available.

Q: What are some examples?

- **Manufacturing Systems.**
  Given a discrete-event simulation model of a semiconductor manufacturing fabrication facility (a fab), one might be interested in maximizing throughput (number of completed wafers) while minimizing average cycle time (mean total time a wafer spends in the fab).
- **Supply Chains.**
  Given a simulation model of a PC manufacturer's supply chain, how can one operate (or possibly reconfigure) the system in order to reduce overall inventory levels and increase customer service levels (response time, fill rate, etc.)?
- **Call Center.**
  Given a simulation model of a complicated call center, how can one operate (or possibly reconfigure) the system in order to minimize system costs (e.g., reduce the number of operators) and increase customer service levels (e.g., reduce waiting times)?
- **Financial.**
  Portfolio Optimization: maximize expected return of a portfolio of financial instruments subject to an acceptable risk level.
  Pricing and Hedging of Financial Derivatives: find

the arbitrage-free price and provide a means to hedge.

- **Single-Server Queue.**
Minimize mean waiting time, assuming a cost on server speed. Let θ denote the mean service time of the server (so $1/\theta$ corresponds to the server speed), and W denote the waiting time. The objective function is the following:

$$J(\theta) = E[W(\theta)] + c/\theta,$$

where c is the cost coefficient for server speed, i.e., a higher-skilled worker costs more. Since W is increasing in θ, the objective function quantifies a trade-off between customer service level and the cost of providing service. For example, one might imagine this as corresponding to the selection of an outdoor ATM, where a faster machine costs more to operate (we are neglecting the initial cost of the machine, since this is a long-run problem). This queueing system is perhaps the most commonly used textbook example for introducing the main components of event-driven simulation (e.g., Law and Kelton 2000).
- (s, S) **Inventory Control System.**
The inventory level of a single item is to be controlled based on two parameters to be optimized, s and S, corresponding to the re-order level and order-up-to level, respectively. When the inventory position falls below s, an order is placed for an amount that would bring the position back up to S. Optimization is generally carried out by minimizing a total discounted or average cost function consisting of ordering, holding, and shortage components.

One can see that there are optimization problems in both the *design* and *operation* of many different types of systems that are modeled using stochastic simulation models. The first four examples are real-world motivating examples, whereas the last two examples are simplified academic problems (renowned OR models in their own right in queueing and inventory theory, respectively), useful for illustrating and testing procedures. For the simplest $M/M/1$ queue in steady state, the single-server optimization problem is analytically tractable, and thus has served as an easy test case for simulation optimization continuous-variable search procedures. Similar remarks are true for the (s, S) inventory system, which is the simplest multi-dimensional problem (as opposed to the previous scalar one), with a nice graphical representation for search procedures. In addition, the parameters may be specified as either continuous or discrete. As a result, it has been used as a test case for nearly all the procedures in the research literature discussed in the next section, e.g., stochastic approximation, sequential response surface methodology, statistical ranking and selection, and multiple comparisons.

Q: What is the general problem setting?

A: As in any optimization problems, there are the usual primary components:

- input and output variables;
- objective function;
- constraints.

The objective function and constraints can involve both the input and output variables, and either (or both) can involve stochastic components. Since the output variables are simulation model performance measures, they are quantitative in nature. However, unlike standard mathematical programs, the input "variables" may be either quantitative or qualitative. For example, in the call center example, one might be deciding between different queue disciplines, e.g., first come, first served, versus a priority scheme. For quantitative input variables, one distinguishes between the continuous values and discrete values, and in the discrete case, between a large state space (uncountable, countably infinite, or just combinatorially large) and a relatively small one. In the latter case, the optimization problem is reduced to an exhaustive comparison of candidate solutions, for which ranking and selection methods are particularly suited.

In general, there is a single objective function. Multiple performance measures are usually handled by combining them into the objective function using appropriate weights, or by including them as constraints. For example, in an inventory problem, one must consider ordering, holding, and backlogging or lost sales. This is usually addressed in one of two (ultimately equivalent) ways: by minimizing a single cost function that has all of these components; or by minimizing a cost function consisting of ordering and holding costs, subject to a service level constraint on lost sales or backlogging.

Constraints can be further subdivided into explicit versus implicit, and deterministic versus stochastic. An explicit constraint is typically something like "the number of operators at location ABC of the call center system cannot exceed 100", whereas an implicit constraint would be more like "the number of operators in the entire call center system (summed over all locations) cannot exceed 1000". These are deterministic constraints, whereas a stochastic constraint might be something like "the proportion of customers having to wait more than one minute for an operator should not exceed 1%", since the waiting time distribution is an output performance measure that must be estimated from the simulation model of the call center.

Q: What distinguishes simulation optimization from "ordinary" optimization?

A: In general, optimization refers to the deterministic domain, and is dominated by techniques such as linear programming, (mixed) integer programming, nonlinear programming, evolutionary algorithms, genetic algorithms, tabu search, and simulated annealing. The key difference in my mind is a point mentioned earlier:
precise evaluation of the objective function is computationally very costly!
This sets up a dichotomy not present in deterministic optimization: that of the search process versus the evaluation process. In other words, simulation optimization involves two important parts: generating candidate solutions and estimating their objective function value. This issue is more fully discussed in Fu (2001b).

Q: What makes simulation optimization difficult?

A: In short, it is the estimation expense derived from the stochastic nature of a complex simulation model that makes simulation optimization doubly difficult on top of the ordinary deterministic optimization setting. 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."

Q: Are you leaving out anything?

A: I am not including a class of problems under the domain of stochastic control, stochastic dynamic programming, or Markov decision processes. Many of these types of problems can be converted into the simulation optimization problem setting, but that is beyond the scope of this tutorial.

2 APPROACHES

Q: What are the main approaches?

A: Banks et al. (2000, pp.488-489) categorize the approaches according to algorithms that

- 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);
- are based on robust heuristics (mainly combinatorial search algorithms that follow evolutionary strategies, e.g., genetic algorithms).

Q: What types of techniques are used?

A: I divide this into the following main categories:

- statistical procedures: sequential response surface methodology, ranking & selection procedures, and multiple comparison procedures;
- metaheuristics: methods directly adopted from deterministic optimization search strategies, such as simulated annealing, tabu search, and genetic algorithms;
- stochastic optimization: random search, stochastic approximation;
- others, including ordinal optimization and sample path optimization.

The remainder of this section summarizes many of these procedures. Much of the material (with the exception of the deterministic search strategies) is condensed from Section 3 in Fu (2001b).

2.1 Ranking and Selection

Ranking & selection procedures and multiple comparison procedures (Goldsman and Nelson 1998, Bechofer, Santner, and Goldsman 1995, and Hochberg and Tamhane 1987) are designed for distinguishing among a *given* set of alternatives. In that sense, they are not simulation optimization procedures, per se, since they lack the search feature. However, there has been recent work combining them with search procedures to yield a more complete optimization package (e.g., Boesel, Nelson, and Ishii 2001, Boesel, Nelson, and Kim 2001). Scenario Seeker (Boesel 1999) uses a heuristic search algorithm, with efficient allocation of simulation replications incorporated into the search phase. Statistical validity for the offered solution are provided using initial screening via subset selection to reduce a possibly large set of configurations to a more manageable size, followed by a standard two-stage ranking & selection procedure to select the best.

2.2 Ordinal Optimization

The key idea behind ordinal optimization (Ho et al. 1992, 2000) is that it is much easier to approximately sort out *relative order* than to precisely estimate (absolute) value. Monte Carlo estimation is limited by the canonical $1/\sqrt{n}$ convergence rate (where n is the number of simulation replications), whereas the probability of correctly selecting the best among a set of alternatives often exhibits convergence rates that are asymptotically exponential ($1-e^{-\Lambda n}$, for some constant Λ). Additional significant computational savings can be achieved by *goal softening*: instead of looking for the best, one settles for a solution that is *good enough*, a term that is statistically defined.

2.3 Stochastic Approximation

Stochastic approximation (SA) mimics the gradient search method from deterministic optimization, but in a rigorous statistical manner that takes into account the stochastic nature of the system model. The general SA algorithm takes the following iterative form (for a minimization problem):

$$\theta_{n+1} = \Pi_\Theta\left(\theta_n - a_n \widehat{\nabla} J(\theta_n)\right),$$

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 $\widehat{\nabla} J$ 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:

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

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

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. Under appropriate conditions, one can guarantee convergence to the actual minimum, 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 (strong) convergence, 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, and 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.

The convergence rate of SA is dramatically enhanced with the availability of direct gradients, one motivating force behind the flurry of research in gradient estimation techniques in the 1990s (e.g., Fu and Hu 1997 and Pflug 1996). The most well-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 gradient estimate based on estimates $\hat{J}$ of the objective function would be given by

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

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

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

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:

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

where $\Delta = [\Delta_1 ... \Delta_p]$ represents a vector of i.i.d. *random* perturbations satisfying certain conditions, which precludes some of the more common continuous distributions such as the normal distribution. The most commonly used perturbation distribution is a 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 an 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 (inexpensive) random perturbations that varies.

2.4 Response Surface Methodology

The goal of response surface methodology (RSM) is to obtain an approximate functional relationship between the input variables and the output objective function. When this is done on the entire (global) domain of interest, the result is often called a metamodel. 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 estimate gradient direction). Again, regression and neural networks are the two most common approaches.

We outline a simple two-stage version of sequential RSM using regression. Phase I involves an iterative gradient search procedure by which a set of points around the current point are simulated (e.g., a 2^p factorial design, where p is the dimension of the input vector), 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 point in the iteration. This process is repeated until the linear fit is deemed inadequate, which signals the end of Phase I. In Phase II, additional points are simulated in the surrounding region of the current point, and a higher order (usually quadratic) regression is carried out to estimate the optimum from the resulting fit.

2.5 Random Search

Random search algorithms move iteratively from a current single design point to another design point in the neighborhood of the current point. A central part of the algorithm is defining an appropriate neighborhood structure, which must be connected in a certain precise mathematical sense. These have been applied primarily to discrete optimization problems, although in principle they could be applied to continuous optimization problems, as well. Differences in algorithms manifest themselves in two main fashions: (a) how the next point is chosen; and (b) what is the estimate 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.

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 point $\hat{\theta}_*$;
     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 pre-specified probability distribution.
     Perform simulations to obtain estimates $\hat{J}(\hat{\theta}_*)$ and $\hat{J}(\theta_i)$.
     Increase counter for point with best estimate and update current point: (**1** denotes indicator function)
         $n_{\hat{\theta}_*} = n_{\hat{\theta}_*} + \mathbf{1}\{\hat{J}(\hat{\theta}_*) \le \hat{J}(\theta_i)\}$;
         $n_{\theta_i} = n_{\theta_i} + \mathbf{1}\{\hat{J}(\hat{\theta}_*) > \hat{J}(\theta_i)\}$;
         If $\hat{J}(\hat{\theta}_*) > \hat{J}(\theta_i)$, then $\hat{\theta}_* \leftarrow \theta_i$.
(3)  Final Answer:
     When stopping rule satisfied, return
         $\theta^* = \arg\max_{\theta \in \Theta} n_\theta$.
```

2.6 Deterministic Optimization Search Strategies

As we shall see in the next section (Table 1), the software implementations are dominated by routines that simply adapt search strategies (mainly evolutionary) from deterministic optimization. Here we will provide a brief overview of three of these approaches: simulated annealing, genetic algorithms (GAs), and tabu search. All of these are global search strategies. However, as anyone who has implemented these algorithms in real problems know, the devil is in the details, in that small tweakings can sometimes lead to vast improvements in performance of the algorithm, so the exposition here will merely provide a flavor for the main ideas.

Simulated annealing (Kirkpatrick, Gelatt, and Vwecchi 1981) can be thought of as a variation of local search (for deterministic objective functions), in which the main idea is to accept all downhill (assume here a minimization problem) improving moves, but *sometimes* accept uphill moves, where the acceptance probability decreases to 0 at an appropriate rate (this is the cooling schedule from which the method derives its name in analogy with the physical annealing process where the system seeks the lowest energy state). An attractive property of this algorithm is that unlike the next two metaheuristics to be described, convergence can be rigorously proven in many settings. On the other hand, in practice, the procedure has been found to be relatively slow in converging to good solutions, compared to the metaheuristic approaches. See Anandalingam (2001) for more details and references.

Tabu search (Glover and Laguna 1997) can be thought of as a variation on local search that incorporates two main strategies: adaptive memory and responsive exploration. The features of these strategies modify the neighborhood of a solution point as the search progresses, and thus determine the effectiveness of the algorithm. In particular, the modification from which the method derives its name forbids certain points (classifying them *tabu*) from belonging to the current neighborhood of points being considered. Thus, for example, short-term memory can prevent the search from revisiting recently visited points, whereas longer-term memory can encourage moves that have historically led to improvements (intensification) and moves into previously unexplored regions of the search space (diversification). See Glover (2001) for more details and references.

Evolutionary search strategies such as GAs work with a *family* of solutions (called the *population*) rather than a single point, as in simulated annealing, random search, and stochastic approximation (and in some ways, sequential RSM, as well). More importantly, the members of the population *interact* in forming the next set of iterates (generation). Otherwise, one could accomplish the same thing using the single point strategies merely by picking a set of starting points and running them in parallel. Here is an outline of a general evolutionary search strategy:

```
(0)  Initialize population.
(1)  Iterate:
       Evaluate fitness of individuals in current generation.
       Choose individuals for reproduction.
       Apply genetic operators to current generation.
       Select new generation.
(2)  Final Answer:
       When stopping rule satisfied, return best individual(s).
```

The fitness of an individual corresponds to the objective function value of a solution point. Important components affecting the success of the algorithm are the selection procedure and the types of genetic operators that are applied. Selection can be done either deterministically or probabilistically, based on the fitness of the individuals. Two of the simplest (deterministic) selection procedures include keeping each generation at a constant number of the fittest individuals (survival of the fittest), or keeping only the offspring from reproduction (complete generational turnover). Genetic operators operate on a genetic representation (code) of the individual, and are generally classified into type main categories: *crossover* (or recombining) operators that involve parents exchanging genetic material, and *mutation* operators that involve only the genetic material of a single individual (this could be local search, for example). Thus, the crossover operators are what distinguish these algorithms from all of the other approaches discussed thus far. See Michalewicz and Schoenauer (2001) for more details and references.

Q: What are the theoretical results for these various procedures?

- Stochastic Optimization Procedures (e.g., SA and random search):

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

  which is also known as almost sure (a.s.) convergence. There are results for other modes of convergence, as well.
- Ranking and Selection Procedures: probability that the selected θ is within ϵ of the best is at least $(1\text{-}\alpha)$.

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

3 SOFTWARE

Q: What kind of software is available?

A: A sample is given in Table 1 (adapted from Law and Kelton 2000, p.664, Table 12.11). All of these software packages came into existence just in the past decade.

Q: What skills does one need to use the software?

A: I'll invoke the consultant's pat answer here: It depends. Most of these packages have been made extremely user "friendly", in the sense that you don't have to know much to be able to use them. However, to be a more informed user, it would be desirable to have the usual skill set in understanding basic simulation output analysis (elementary probability and statistics), as well as a rudimentary knowledge of the optimization approaches that are used by the algorithms. I'll provide a bit more details on three of the packages listed in Table 1.

The optimization routine in the AutoStat suite (Bitron 2000) of statistical output analysis tools incorporates an evolutionary strategies algorithm (genetic algorithm variation) and handles multiple objectives by requiring weights

Table 1: Some 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

to form a fitness function. For each input variable the user wishes to optimize, 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. 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.

SIMUL8's OPTIMIZ proceeds using a form of sequential RSM using neural networks (`http://www.SIMUL8.com/optimiz1.htm` July 11, 2001):

> "SIMUL8 OPTIMIZ searches for the best solution. Give OPTIMIZ information about what to optimize (maybe a service level of 95%). Give it a list of the resources and other variables you are prepared to see change (maybe some factors are fixed but you could buy more machinery, or some types of labor). You can also give constraints on how much these factors are allowed to change.
>
> 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 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."

OptQuest is a stand-alone optimization software routine that can be bundled with a number of the commercial simulation environments, such as Arena and 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. Scatter search is also a population-based evolutionary search strategy like GAs. However, Glover, Kelly, and Laguna (1999) claim that whereas naïve GAs 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 neural network serves as a metamodel representation. Since it 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.

Q: Why is the commercial software dominated by metaheuristic approaches?

A: Good question, especially considering that SA and RSM have been around half a century. I don't think there is a clear answer to this, and Fu (2001b) contains further discussion on this topic. Some possible explanations include the fact that the metaheuristic approaches use a family of solutions, are designed to seek global optimality (explore solutions over the entire state space), and seem to have robust properties in practice, even if not completely supported theoretically yet. Some of the other techniques, e.g., the PA gradient estimation technique, are local search strategies and can be problem dependent, so they are more apt to be used in specific consulting problems (such as those cases reported in Kapuscinski and Tayur 1999); however, these techniques, when they apply, are usually more effective and efficient. Closely related to this point is that these techniques may simply be harder (but certainly not insurmountable) to program into general-purpose simulation environments.

4 CONCLUSION

Q: Where do I go from here?

- **Research Basics.** From the reference list provided here, good places to start include the surveys by Fu (1994, 2001a, 2001b), Andradóttir (1998), and Swisher et al. (2001), as well as Chapter 12 in Law and Kelton (2000) and Banks et al. (2000), and the book by Pflug (1996). Other good sources include these annual Winter Simulation Conference Proceedings and the forthcoming *ACM Transactions on Modeling and Computer Simulation* Special Issue on Simulation Optimization (co-edited by myself and Barry Nelson).
- **Newer Avenues (>1990).** These include SPSA (Spall 1992, Fu and Hill 1997, Kushner and Yin 1997), ordinal optimization (Ho et al. 1992, 2000; Chen et al. 2000), sample path optimization (Gürkan, Özge, and Robinson 1999), nested partitions (Shi and Olafsson, 2000), ant colony optimization (Dorigo and Di Caro 1999), neuro-dynamic programming (Bertsekas and Tsitsiklis 1996).
- **Practice.** See the software vendors peddling their wares at the WSC!

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grants DMI-9713720 and DMI-9988867, and by the Air Force of Scientific Research under Grant F496200110161.

REFERENCES

Anandalingam, G. 2001. Simulated annealing. In *Encyclopedia of Operations Research and Management Science*, 2nd edition, ed. S. Gass and C. Harris, 748-751. Boston: Kluwer Academic Publishers.

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

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

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

Bertsekas, D.P. and J.N. Tsitsiklis. 1996. *Neuro-Dynamic Programming*. Belmont: Athena Scientific.

Bitron, J. February 2000. Optimizing AutoMod Models with AutoStat. *AutoFlash* Monthly Newsletter. published by AutoSimulations.

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

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

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

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

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

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

Fu, M.C. 2001a. Simulation optimization. In *Encyclopedia of Operations Research and Management Science*, 2nd edition, ed. S. Gass and C. Harris, 756-759. Boston: Kluwer Academic Publishers.

Fu, M.C. 2001b. Optimization for simulation: Theory vs. Practice (Feature Article). *INFORMS Journal on Computing* forthcoming.

Fu, M.C., S. Andradóttir, J.S. Carson, F. Glover, C.R. Harrell, Y.C. Ho, J.P. Kelly, and S.M. Robinson, 2000. Integrating optimization and simulation: research and practice. In *Proceedings of the 2000 Winter Simulation Conference*, eds. J.A. Joines, R.R. Barton, K. Kang, and P.A. Fishwick, 610-616. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers. Available online via `http://www.informs-cs.org/wsc00papers/082.PDF`.

Fu, M.C. and S.D. Hill. 1997. Optimization of discrete event systems via simultaneous perturbation stochastic approximation. *IIE Transactions* 29 (3): 233-243.

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

Glover, F. 2001. Tabu search. In *Encyclopedia of Operations Research and Management Science*, 2nd edition, ed. S. Gass and C. Harris, 821-827. Boston: Kluwer Academic Publishers.

Glover, F., J.P. Kelly, M. Laguna. 1999. New advances for wedding optimization and simulation. In *Proceedings of the 1999 Winter Simulation Conference*, eds. J.A. Joines, R.R. Barton, K. Kang, and P.A. Fishwick, 255-260. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

Glover, F. and M. Laguna. 1997. *Tabu Search.* Boston: Kluwer Academic.

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

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

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

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

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

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

Kirkpatrick, S., C.D. Gelatt, and M.P. Vwecchi. 1981. Optimization by simulated annealing. *Science* 220: 671-680.

Kleijnen, J.P.C. 1998. Experimental design for sensitivity analysis, optimization, and validation of simulation models. Chapter 6 in *Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice*, ed. J. Banks. New York: John Wiley & Sons.

Kushner, H.J. and G.G. Yin. 1997. *Stochastic Approximation Algorithms and Applications*. New York: Springer-Verlag.

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

Michalewicz, Z. and M. Schoenauer. 2001. Evolutionary algorithms. In *Encyclopedia of Operations Research and Management Science*, 2nd edition, ed. S. Gass and C. Harris, 264-269. Boston: Kluwer Academic Publishers.

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

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

Spall, J.C. 1992. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE Transactions on Automatic Control* 37 (3): 332-341.

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

AUTHOR BIOGRAPHY

MICHAEL C. FU `<mfu@rhsmith.umd.edu>` is a Professor in the Robert H. Smith School of Business, with a joint appointment in the Institute for Systems Research and an affiliate appointment in the Department of Electrical and Computer Engineering, all at the University of Maryland. He received degrees in mathematics and EE/CS from MIT, and a Ph.D. in applied mathematics from Harvard University. His research interests include simulation and applied probability modeling, particularly with applications towards manufacturing systems, inventory control, and financial engineering. He teaches courses in applied probability, stochastic processes, simulation, computational finance, and operations management, and in 1995 was awarded the Maryland Business School's Allen J. Krowe Award for Teaching Excellence. He is a member of **INFORMS** and **IEEE**. He is currently the Simulation Area Editor of *Operations Research*, and serves on the editorial boards of *Management Science*, *IIE Transactions*, and *Production and Operations Management*. He is also serving as Guest co-Editor for a special issue on simulation optimization for the *ACM Transactions on Modeling and Computer Simulation*. He is co-author (with J.Q. Hu) of the book, *Conditional Monte Carlo: Gradient Estimation and Optimization Applications*, which received the INFORMS College on Simulation Outstanding Publication Award in 1998.