TUTORIALS IN
OPERATIONS RESEARCH
INFORMS 2010

 | ISBN 978-0-9843378-0-4
DOI 10.1287/educ.1100.0069

Optimization via Simulation Over Discrete Decision Variables

Barry L. Nelson
Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, Illinois 60208, nelsonb@northwestern.edu

Abstract Both the simulation research and software communities have been interested in optimization via simulation (OvS), by which we mean maximizing or minimizing the expected value of some output of a stochastic simulation. Continuous-decision-variable OvS, and gradient estimation to support it, has been an active research area with significant advances. However, the decision variables in many operations research and management science simulations are more naturally discrete, even categorical. In this tutorial we describe some of the research directions and results available for discrete-decision-variable OvS, and provide some guidance for using the OvS heuristics that are built into simulation modeling software.

Keywords simulation; stochastic optimization; random search; ranking and selection

1. Introduction by Example

This tutorial covers a class of problems that may be solved by *optimization via simulation* (OvS). There is not universal agreement on the name "optimization via simulation" for the problems discussed here, with "simulation optimization" and "optimization for simulation" often used to mean the same thing. To build some intuition, we start with a collection of OvS examples.

Designing a Highly Reliable System

A system works only if all of its subsystems work; the subsystems consist of components that have their own time-to-failure and repair-time distributions. The objective is to decide how many and what redundant components to use to minimize steady-state system unavailability given budget constraints. The budget is relatively tight, so altogether there are only 152 feasible configurations. Let $x \in \{1, 2, \ldots, 152\}$ index the configurations.

Flow-Line Throughput

A three-stage flow line has finite buffer storage space in front of stations 2 and 3 (the number of spaces being denoted by x_4 and x_5) and an infinite number of jobs in front of station 1. There is a single server at each station, and the service-time distribution at station i has service rate x_i, $i = 1, 2, 3$. If the buffer of station i is full, then station $i-1$ is blocked and a finished job cannot be released from station $i-1$. The total buffer space and the service rates are limited by constraints on space and cost. The objective is to find a buffer allocation and service rates such that the expected throughput over a one-year planning horizon is maximized. The constraints are $x_1 + x_2 + x_3 \leq 20$, $x_4 + x_5 = 20$, $1 \leq x_i \leq 20$, and $x_i \in \mathcal{Z}^+$ for $i = 1, 2, \ldots, 5$, implying 21,660 feasible solutions. This example is adapted from Buzacott and Shantikumar [9] and Pichitlamken and Nelson [27].

Inventory Management with Dynamic Customer Substitution

A retailer faces a one-shot inventory stocking decision for six product variants at the beginning of the selling season so as to maximize the expected value of profit. No inventory replenishment can occur, and there is no salvage value for the products. Each consumer selects the available product with the highest utility for them, which may be a no-purchase option. The number of customers is Poisson, and the customer's choice behavior is modeled by a multinomial logit model. Pricing is an exogenous decision. Let $x_1, x_2, \ldots, x_6$ denote the number of each variant the retailer chooses to stock. Taking into account upper bounds of 500 on the number ordered for each variant, there are $500^6 \approx 1.6 \times 10^{16}$ feasible solutions. This example is adapted from Mahajan and Van Ryzin [20].

These three problems share the following characteristics: The system of interest is stochastic, the objective is to optimize a performance measure that is the mathematical expected value of a stochastic output (unavailability, throughput, and cost, respectively), and the system is complex enough that this expected value is neither analytically nor numerically tractable but it can be estimated by running a simulation. The number of feasible solutions ranges from modest to monstrous, but in all cases it is finite. Discrete optimization via simulation (DOvS) addresses these types of problems (as well as problems with a countable number of feasible solutions, which will not be covered here).

To be precise, the problems considered in this tutorial are of the form

$$\min\{g(\mathbf{x}) = E_{\mathbf{x}}[Y(\mathbf{x})]\}, \tag{1}$$

$$\mathbf{x} \in \Theta = \Phi \cap \mathcal{Z}^d, \tag{2}$$

$$E_{\mathbf{x}}[C_i(\mathbf{x})] \le q_i, \quad i = 1, 2, \ldots, s. \tag{3}$$

In all cases, the distribution of the random variable $Y(\mathbf{x})$, as a function of the decision variable $\mathbf{x}$, is unknown, but realizations of $Y(\mathbf{x})$ can be observed through simulation experiments at setting $\mathbf{x}$. If the problem is more naturally thought of as maximization, we can always formulate an equivalent minimization version.

In the first part of the tutorial, $\mathbf{x}$ is just an index in the set $\{1, 2, \ldots, k\}$, and the solutions may be categorical (not ordered in any way). In the second part, $\mathbf{x}$ is a vector of d integer-ordered decision variables in a feasible region $\Phi \subset \Re^d$, possibly defined by a set of constraints, where $\mathcal{Z}^d$ denotes all d-dimensional vectors with integer components. In this case, we typically assume that Φ is compact and convex.

The constraints (3) are on the expected values of additional output performance measures, $C_1(\mathbf{x}), C_2(\mathbf{x}), \ldots, C_s(\mathbf{x})$. For instance, in the flow-line problem $C_1(\mathbf{x})$ might be the overall system work in process, which we would like to keep below a certain level q, on average. Stochastic constraints have not yet received much attention in the DOvS literature, so we focus primarily on problems where there are no such constraints and comment on stochastic constraints only briefly.

We always assume that $\mathrm{Var}[Y(\mathbf{x})] < \infty$ for all $\mathbf{x} \in \Theta$, and that we can simulate independent and identically distributed (i.i.d.) replications, $Y_1(\mathbf{x}), Y_2(\mathbf{x}), \ldots$, at any $\mathbf{x}$ (with an analogous assumption for any $C_i(\mathbf{x})$).

Notice that we are not including problems for which $\mathbf{x} \in \Phi$, that is, continuous-valued decision variables. This is an important area of research that has received much study; see, for instance, Barton and Meckesheimer [4] and Fu [11].

The organization of this tutorial is as follows: We first comment on the gap between OvS research and practice. We then discuss ranking-and-selection (R&S) techniques, which apply to DOvS problems where $|\Theta|$ is relatively small. Next we present some key approaches for adaptive random search (ARS), the primary provably convergent methodology for large $|\Theta|$. Some recent results on working with stochastic constraints are provided in §5. We finish by describing a public domain DOvS solver and giving some tips on working with the commercial solvers integrated into simulation software.

2. OvS Practice and Research

Nearly every commercial discrete-event or spreadsheet simulation software package has a built-in OvS solver. In the simulation software world, the modeling environment is the primary point of competition and distinction. As a result, the modeling environments are all different, and the OvS solver must be customized to work with the specific simulation software. This contrasts, with a few notable exceptions, to the mathematical programming world, where the solver is the point of competition and there is relative agreement as to what constitutes a problem formulation.

There are at least two stand-alone OvS solvers, one commercial (OptQuest, http://www.opttek.com) and one open source (Industrial Strength COMPASS, http://www.iscompass.net). In either case the user is responsible for building a connection between the solver and the simulation, which is probably easiest when the simulation is programmed in a lower-level language like `C++` or `Java` rather than a commercial modeling environment.

As has been well documented by Fu [10], the commercial OvS solvers have not been strongly influenced by academic research on OvS. The commercial solvers are driven by metaheuristics, such as tabu search and genetic algorithms, that have proven to be effective on difficult, mixed-decision-variable deterministic optimization problems (see Ólafsson [26] for a review of metaheuristics for OvS). As a result, they do not provide the sort of convergence guarantees that are prized in the research literature. However, this gap is beginning to close, and convergent algorithms that can solve real problems are starting to appear, so this tutorial will emphasize the research literature that is most likely to contribute to eliminating the gap. We also provide suggestions about how to effectively use the current commercial solvers.

The research we will cover addresses algorithms that either exhaust Θ—that is, simulate every feasible solution—or are based on some form of ARS. In deterministic mathematical programming, exhausting Θ guarantees finding the optimal solution; this is not true in stochastic simulation because we can only estimate the value of the objective function, and therefore never know it for sure. In ARS, solutions are randomly sampled from Θ, and the algorithms are "adaptive" in how they use observed performance to decide which solutions are eligible to be sampled and how much simulation effort is expended on them.

A useful, although not exhaustive, way to categorize DOvS research is as follows (this is based partially on Andradóttir [2]):

Solution sampling: Does the algorithm use point-based (from one solution to the next), set-based (from solutions in an eligible set), or population-based (combining components from a set of solutions) sampling of solutions? To be comprehensive, we include exhausting $\mathbf{x} \in \Theta$ as a form of population-based sampling. Most ARS algorithms are set based.

Type of simulation: Does the algorithm work for finite-horizon or infinite-horizon (steady-state) simulation, or both? In infinite-horizon simulation, the stopping time of the simulation replication is an experimental factor, along with the number of replications; whereas in finite-horizon simulation, only the number of replications matters (the stopping time of the simulation is fundamental to the definition of the problem). In this tutorial we will only consider the number of replications as an experimental factor, which means that we are either optimizing a finite-horizon simulation or an infinite-horizon simulation for which the run length has already been chosen.

Feasible region: Is Θ finite or countably infinite; is it integer ordered or combinatorial? Our focus is finite and small enough to exhaust, or finite but large and integer ordered.

Guarantee: Does the algorithm guarantee almost-sure (probability 1) convergence to a globally or locally optimal solution, or guarantee a probability of correctly selecting the optimal solution, or none of the above?

3. Ranking and Selection

R&S addresses problems for which we can exhaust Θ and we want some statistical guarantee about the quality of the solution $\mathbf{x}$ that we select after simulating them all. R&S prescribes an

experimental design and statistical estimators that provide the desired terminal guarantee. R&S procedures were invented in the 1950s for statistical selection problems such as choosing the best treatment for a medical condition, but they have probably found their greatest application in DOvS. There are two basic paradigms in R&S: frequentist and Bayesian. Although we focus on the frequentist approach, we give a brief overview of the Bayesian perspective.

There are two streams of Bayesian R&S procedures: the optimal computing budget allocation (OCBA) procedures and expected value of information (EVI) procedures (see Branke et al. [8] for an excellent overview and comparison with frequentist procedures). The OCBA procedures allocate the simulation budget to maximize the posterior probability of correct selection (PCS), where a "correct selection" means choosing the solution $\mathbf{x}$ with the smallest $g(\mathbf{x})$. The EVI procedures and their extensions try to minimize the expected opportunity cost of the chosen solution. A strength of the Bayesian procedures is that they provide strategies for optimally allocating simulation effort to the solutions being considered.

Next we introduce the frequentist approach. Suppose that there are $k \geq 2$ solutions in Θ, denoted as $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k$. Let $Y_j(\mathbf{x}_i)$ denote the jth i.i.d. observation from simulating solution $\mathbf{x}_i$. A common assumption is that $Y_j(\mathbf{x}_i) \sim N(g(\mathbf{x}_i), \sigma^2(\mathbf{x}_i))$, where $g(\mathbf{x}_i)$ and $\sigma^2(\mathbf{x}_i)$ are unknown. Without loss of generality, we let $g(\mathbf{x}_1) \leq g(\mathbf{x}_2) \leq \cdots \leq g(\mathbf{x}_k)$, so the goal of an R&S procedure is to select solution $\mathbf{x}_1$ (which is unknown to the user). An R&S procedure is statistically valid if

$$\Pr\{\text{select solution } \mathbf{x}_1 \mid g(\mathbf{x}_1) \leq g(\mathbf{x}_2) - \delta\} \geq 1 - \alpha, \tag{4}$$

where $\delta > 0$ is called the indifference-zone (IZ) parameter and is typically set to the smallest difference that is practically significant. Under this formulation, the best solution $\mathbf{x}_1$ will be selected with a probability at least $1 - \alpha$ as long as the difference between the objective values of the best and second-best solutions is large enough to matter. If there are a set of solutions whose objective values are within δ of the objective value of the best, then all solutions in that set are acceptable. Some R&S procedures also guarantee to select one of the acceptable solutions with probability at least $1 - \alpha$ when there is more than one (Nelson and Goldsman [22]).

R&S originally addressed problems where the number of feasible solutions k was assumed quite small, say, less than 20. To tackle problems like the design of the highly reliable system, which had $k = 152$ feasible solutions, enhancements are needed; this has been a focus of the simulation research community. Here, we give one example: procedure NSGS (Nelson et al. [23]). NSGS is valid if the output from each feasible solution is i.i.d. normal and is independent across different solutions. NSGS is the combination of a subset-selection procedure that tries to eliminate noncompetitive solutions without expending much simulation effort, and a ranking procedure applied to the competitive systems remaining in the subset.

3.1. Procedure NSGS

(1) Specify the overall desired probability of correct selection $1/k < 1 - \alpha < 1$, the IZ parameter $\delta > 0$, a common initial number of replications from each system $n_0 \geq 2$, and the initial number of competing systems $k = |\Theta|$. Furthermore, set

$$t = t_{n_0 - 1,\, (1-\alpha/2)^{1/(k-1)}}$$

the $(1-\alpha/2)^{1/(k-1)}$ quantile of the t distribution with $n_0 - 1$ degrees of freedom, and obtain Rinott's constant $h = h(n_0, k, 1 - \alpha/2)$ from the tables in Wilcox [33], Bechhofer et al. [6], or Goldsman and Nelson [12].

(2) Take n_0 replications from each feasible solution. Calculate the first-stage sample means $\bar{Y}(\mathbf{x}_i; n_0)$ and marginal sample variances

$$S(\mathbf{x}_i)^2 = \frac{1}{n_0 - 1} \sum_{j=1}^{n_0} (Y_j(\mathbf{x}_i) - \bar{Y}(\mathbf{x}_i; n_0))^2$$

for $i = 1, 2, \ldots, k$.

(3) *Subset Selection.* Calculate the quantity

$$W_{ij} = t\left(\frac{S(\mathbf{x}_i)^2 + S(\mathbf{x}_j)^2}{n_0} \right)^{1/2}$$

for all $i \neq j$. Form the screening subset I, containing every feasible solution $\mathbf{x}_i$ such that $1 \leq i \leq k$ and

$$\bar{Y}(\mathbf{x}_i; n_0) \leq \bar{Y}(\mathbf{x}_j; n_0) + W_{ij} \quad \text{for all } j \neq i.$$

(4) If $|I| = 1$, then stop and return the system in I as the best. Otherwise, for all $\mathbf{x}_i \in I$, compute the second-stage sample sizes

$$N_i = \max\{n_0, \lceil (hS(\mathbf{x}_i)/\delta)^2 \rceil\},$$

where $\lceil \cdot \rceil$ is the ceiling function.

(5) Take $N_i - n_0$ additional replications from all systems $\mathbf{x}_i \in I$.

(6) Compute the overall sample means $\bar{Y}(\mathbf{x}_i; N_i)$ for all $\mathbf{x}_i \in I$. Select the system $\mathbf{x}_B = \arg\min_{\mathbf{x}_i} \bar{Y}(\mathbf{x}_i; N_i)$ as best.

NSGS guarantees that $\mathbf{x}_B = \mathbf{x}^*$ or $g(\mathbf{x}_B)$ is within δ of $g(\mathbf{x}^*)$, and also that $g(\mathbf{x}_B) \in \bar{Y}(\mathbf{x}_B; N_B) \pm \delta$, all with probability $\geq 1 - \alpha$.

NSGS has been applied to problems with more than 1,000 feasible solutions and tends to be very efficient when there are a few competitive solutions and many noncompetitive ones; this often happens when Θ is obtained by trying every feasible combination of some more basic decision variables. The procedure works, even if this is not the case, but may be computationally expensive if the subset-selection step (3) cannot screen out a significant number of feasible solutions.

In this section we have described R&S procedures as DOvS procedures themselves; however, R&S has also been used in conjunction with other DOvS optimization algorithms to improve their efficiency or to make a correct decision at the end of the optimization process. In particular, Boesel et al. [7] proposed an extension to NSGS that they call a cleanup procedure. The concept of "cleanup" is to take the solutions simulated by a DOvS algorithm, eliminate without any additional simulation the ones that are not statistically competitive, and perform just enough additional simulation on the remaining solutions to provide *a statistical guarantee of selecting the best solution from among those simulated during the search and provide a confidence interval on how good the selected solution is.*

Many search-based OvS algorithms select the best solution from a neighborhood. Pichitlamken et al. [28] designed a sequential R&S procedure for that purpose. Similarly, Xu et al. [34] used the comparison-with-a-standard procedure of Kim [19] to test the local optimality of a solution when solving DOvS problems.

4. Adaptive Random Search

When $|\Theta|$ is large, then ARS procedures dominate the research literature. We consider only the case where $\mathbf{x} \in \Theta$ are integer-ordered decision variables so that there is a natural concept of a neighborhood, although some of the algorithms we describe can also be used on combinatorial problems.

We start with globally convergent algorithms. To simplify the presentation we assume there is a single, globally optimal solution $\mathbf{x}^*$. The algorithms we present also converge when there are multiple global optima, but this complicates the definition of convergence without adding any insight for our purposes. The algorithms we describe have different, but similar, definitions of "global convergence," but informally they all produce a sequence of solutions $\mathbf{x}_k^*, k = 1, 2, \ldots$, that degenerate to $\mathbf{x}^*$ in a very strong sense. One can also talk about convergence of $\hat{g}(\mathbf{x}_k^*)$, the estimated optimal value, but we do not do so here. See, for instance, Andradóttir [1].

An example of a globally convergent random search algorithm for DOvS problems is the stochastic ruler (Yan and Mukai [36]); examining it helps us understand the objectives of simulation researchers when creating better ARS algorithms. In the algorithm below, $\hat{g}(\mathbf{x})$ is an unbiased estimator of $g(\mathbf{x})$, whose distribution need not change as a function of the iteration k; e.g., it could be $\bar{Y}(\mathbf{x}; n)$ with n fixed.

Stochastic Ruler

Step 0. Choose a and b such that $\Pr\{a \le \hat{g}(\mathbf{x}) \le b\} = 1$ for all $\mathbf{x} \in \Theta$, an irreducible Markov chain transition matrix $\mathbf{R}$ on Θ such that $\mathbf{R}(\mathbf{x}, \mathbf{x}') = \mathbf{R}(\mathbf{x}', \mathbf{x})$ for all solutions $\mathbf{x}, \mathbf{x}' \in \Theta$, and a sequence of positive integers m_k such that $m_k \to \infty$ as $k \to \infty$. Select an initial solution $\mathbf{x}_0^*$, and set $k \leftarrow 0$.

Step 1. Generate a candidate solution $\mathbf{x}'$ from $\mathbf{R}(\mathbf{x}_k^*, \cdot)$; in other words, randomly select a solution using the $\mathbf{x}_k^*$ row of $\mathbf{R}$ as the probability distribution on solutions.

Step 2. For $i \leftarrow 1$ to m_k do:

> Generate an independent estimate $\hat{g}(\mathbf{x}')$ of $g(\mathbf{x}')$
> Generate $U \sim U(a, b)$
> If $\hat{g}(\mathbf{x}') > U$, then
> $\mathbf{x}_{k+1}^* \leftarrow \mathbf{x}_k^*$; go to Step 3
> endif
> Next i
> $\mathbf{x}_{k+1}^* \leftarrow \mathbf{x}'$

Step 3. $k \leftarrow k + 1$; go to Step 1.

Why does the stochastic ruler work? First, one can show that the probability of rejection at Step 2 is minimized at $\mathbf{x}^*$, and furthermore, the transition probability into $\mathbf{x}^*$ is greater than out of $\mathbf{x}^*$. In addition, the transition matrix of the implied discrete-time Markov chain is irreducible, aperiodic, and finite, and it can be shown that the steady-state probabilities of the Markov chain degenerate to a distribution putting probability 1 on $\mathbf{x}^*$ as $k \to \infty$.

The algorithm is beautiful and compact (because it retains no past data), but is not really adaptive and requires increasing effort from iteration to iteration. Therefore, the finite-time performance is not good, even though it can be proven to converge. Research in ARS has focussed on improving finite-time performance by being more adaptive—that is, more sensitive to what has been observed—without sacrificing this sort of convergence guarantee.

Andradóttir [1] noted that there are significant advantages to using the cumulative sample mean to estimate the value of the optimal solution, even if it is not used for guiding the ARS. She showed that this approach makes almost-sure convergence of the algorithms easy to prove, provides a better estimate of the true value of the selected solution whenever the algorithm terminates, and tends to yield better empirical performance. Thus, an improved stochastic ruler algorithm would use the same guidance system to select the next $\mathbf{x}'$ to evaluate, but would report as the estimated optimal the one with the smallest sample mean $\bar{Y}(\mathbf{x}; n(\mathbf{x}))$, where $n(\mathbf{x})$ is the cumulative number of replications of solution $\mathbf{x}$ over all times it has been visited. Because it is now computationally possible to store sample mean information on a very large number of solutions, this insight has had a profound impact on algorithm design.

In the following subsections, we examine some important threads in globally convergent ARS research, then consider the concept of locally convergent algorithms.

4.1. Branch and Bound for DOvS

When solving deterministic integer programming problems, branch-and-bound methods that relax integrality constraints are often used, which is possible because the objective function can be evaluated at noninteger values. When solving DOvS problems, however, the simulation may not make sense at fractional values of $\mathbf{x}$. For instance, it is not clear how to simulate a queueing system with 3.6 servers or queue capacity 5.2. And as opposed to having a known linear, quadratic, or even convex objective function, the function $g(\mathbf{x})$ is implied by the simulation model, and little structural information about it is available.

Nevertheless, the principles of branch and bound have had an important influence on DOvS. We start by describing the stochastic branch-and-bound method (SB&B) of Norkin et al. [24, 25], and then show how their reasoning leads to the nested partitions (NP) methods of Shi and Ólafsson [31, 32] and Pichitlamken and Nelson [27], which have seen considerable practical use.

To describe a simplified version of SB&B, let $\{\Theta^p\}$ be subsets of Θ creating a partition $\mathcal{P}$. Define as

$$g^*(\Theta^p) = \min_{\mathbf{x}\in\Theta^p} g(\mathbf{x})$$

the value of the optimal solution restricted to Θ^p. Clearly, $g(\mathbf{x}^*) = \min_{\Theta^p\in\mathcal{P}} g^*(\Theta^p)$. We assume that there exist two bounding functions l and u defined on subsets of Θ such that

- $l(\Theta^p) \le g^*(\Theta^p) \le u(\Theta^p)$,
- $u(\Theta^p) = g(\mathbf{x}')$ for some $\mathbf{x}' \in \Theta^p$,
- if $|\Theta^p| = 1$, then $l(\Theta^p) = g^*(\Theta^p) = u(\Theta^p)$.

Of course, if we knew l and u, then we could directly apply branch and bound. Instead, suppose that there are estimators L_k and U_k defined on subsets Θ^p such that

$$\lim_{k\to\infty} L_k(\Theta^p) = l(\Theta^p),$$
$$\lim_{k\to\infty} U_k(\Theta^p) = u(\Theta^p),$$

with probability 1, where k will be an index of algorithm/simulation effort. Under these assumptions, an SB&B algorithm is as follows:

Stochastic Branch and Bound

(1) Set $k = 0$, $\mathcal{P}_0 = \Theta$ and generate $L_k(\Theta)$ and $U_k(\Theta)$.
(2) Set

$$\Theta_k = \arg\min\{L_k(\Theta^p)\colon \Theta^p \in \mathcal{P}_k\},$$
$$\mathbf{x}_k^* \in \arg\min\{U_k(\Theta^p)\colon \Theta^p \in \mathcal{P}_k\}.$$

(3) If $|\Theta_k| = 1$, then $\mathcal{P}_{k+1} = \mathcal{P}_k$ and go to Step 4.
Else let $\mathcal{P}'_k$ be a partition of Θ_k, and let $\mathcal{P}_{k+1} = (\mathcal{P}_k \backslash \Theta_k) \cup \mathcal{P}'_k$.
(4) For all $\Theta^p \in \mathcal{P}_{k+1}$ generate $L_{k+1}(\Theta^p)$ and $U_{k+1}(\Theta^p)$, set $k = k+1$, and go to Step 2.

In brief, as the algorithm progresses, better and better estimates are obtained of the bounding functions, and the partition with the best lower bound is partitioned finer and finer. It can be shown that $\mathbf{x}_k^*$ converges with probability 1 to $\mathbf{x}^*$ under very reasonable conditions.

There are two obvious drawbacks to direct application of SB&B. First, there needs to be bounding functions l and u and convergent estimators of them. Norkin et al. [24, 25] and Gutjahr et al. [13] display some specific stochastic optimization problems for which this is the case. A second issue is the overhead needed to retain and refine a larger and larger

partition structure as the algorithm progresses, because no partition is ever eliminated from consideration as in deterministic branch and bound.

Clearly, the key is the lower bounding function l. Notice that for any subset Θ^p, it is trivially true that

$$l(\Theta^p) = \min_{\mathbf{x}\in\Theta^p} g(\mathbf{x}),$$
$$L_k(\Theta^p) = \min_{\mathbf{x}\in\Theta^p} \bar{Y}(\mathbf{x}; n(k))$$

satisfy the required conditions provided $n(k)$, the cumulative number of replications of solution $\mathbf{x}$ through iteration k, increases. But it is also true that the estimator

$$\hat{L}_k(\Theta^p) = \min_{\mathbf{x}\in\mathcal{X}^p(k)} \bar{Y}(\mathbf{x}; n(k))$$

works provided $\mathcal{X}^p(k)$ is a randomly sampled subset of solutions from Θ^p and $\mathcal{X}^p(k)$ converges to Θ^p as $k \to \infty$. Thus, $\hat{L}_k(\Theta^p)$ is a sampling-based lower bound that is available for any problem. This addresses the first drawback of SB&B.

To avoid the need to carry along information on an increasing number of partitions, we can modify the definition of the new partition, $\mathcal{P}_{k+1}$, to be

$$\mathcal{P}_{k+1} = (\Theta \backslash \Theta_k) \cup \mathcal{P}'_k.$$

In words, we only maintain the most recently refined partition and aggregate all other solutions into a single "surrounding region."

With these two refinements, we have a version of the NP method similar to Pichitlamken and Nelson [27]. The NP method is relatively easy to implement and does not depend on special structure, although special structure can be exploited and may be particularly helpful in designing a partitioning scheme. The NP method uses a very straightforward adaptation: sample solutions more intensely in the partition that has most recently provided an apparently good solution. As a refinement of both SB&B and NP, Xu [35] suggests using a more sophisticated sampling-based bound than $\hat{L}_k$, while still maintaining the full partitioning structure of SB&B.

4.2. The Global–Local Balance

A goal of globally convergent ARS is to exploit (search intensively) regions of Θ that appear to have good solutions, while maintaining enough global exploration to be sure to capture $\mathbf{x}^*$ in the limit. In addition, because $g(\mathbf{x})$ can only be estimated, there must be less and less estimation error in the limit to facilitate convergence. The following subsections describe two ways to balance these competing objectives.

4.2.1. BEESE. Prudius and Andradóttir [29] proposed the balanced explorative and exploitative search with estimation (BEESE) scheme, which keeps global exploration, local exploitation, and solution estimation in play by switching back and forth among them. Specifically, there is a `Global` probability distribution that places positive probability on all elements of Θ for exploration; a family of `Local` probability distributions that (typically) assigns probability only to solutions that are close (in some sense) to the current sample best solution in Θ for exploitation; and an estimation scheme that allocates replications to a solution $\mathbf{x}$ to estimate $g(\mathbf{x})$. The global convergence of an algorithm that falls into the BEESE framework can be proved if the `Global` search distribution satisfies certain conditions. The simplest version of BEESE, known as Randam BEESE (R-BEESE), has the following high-level structure:

R-BEESE

(1) Sample a solution $\mathbf{x} \sim$ `Global`(Θ) and estimate $g(\mathbf{x})$.

(2) With probability q, take additional replications of the current sample best solution $\hat{\mathbf{x}}^*$ to refine the estimate of $g(bx^*)$.

Else with probability p sample a solution $\mathbf{x} \sim \texttt{Global}(\Theta)$ and estimate $g(\mathbf{x})$ or refine the estimate of $g(\mathbf{x})$ if $\mathbf{x}$ has been visited before.

Otherwise sample solution a $\mathbf{x} \sim \texttt{Local}(\hat{\mathbf{x}}^*)$ and estimate or refine the estimate of $g(\mathbf{x})$.

(3) Update current sample best solution and go to Step 2.

Clearly, the `Global` and `Local` distributions on Θ, and the choice probabilities p and q, have an impact on performance. Prudius and Andradóttir [29] also describe as Adaptive BEESE (A-BEESE), which makes these decisions dynamic and adaptive to the progress of the search. The concept of the BEESE framework is to provide a structure and conditions that guarantee global convergence, but within which smart heuristics can be employed.

4.2.2. MRAS. Hu et al. [18] distinguish between instance-based DOvS search algorithms—where new candidate solutions are generated directly from previous candidates—and model-based DOvS search, where new candidate solutions are sampled from a global probability distribution over Θ that adapts based on the performance of previous candidates. Model-based search is always global but concentrates the search on promising areas by changing the global distribution; this is a different approach to staying global than BEESE, which always returns to global sampling but does not update the global distribution. We briefly describe the model reference adaptive search (MRAS) algorithm of Hu et al. [17, 18] and its OvS counterpart SMRAS, which are closely related to the cross-entropy method (Rubinstein and Kroese [30]) and estimation of distribution algorithm (Mühlenbein and Paaß [21]).

To provide intuition, consider a problem where the goal is maximization, $g(\mathbf{x}) \geq 0$, and g can be evaluated exactly; that is, we have a zero-variance estimator of $g(\mathbf{x})$, so the optimization problem is deterministic. Let $r(\cdot)$ be a probability mass function over $\mathbf{x} \in \Theta$, which defines a random variable $\mathbf{X} \sim r$; in other words, $\mathbf{X}$ is a randomly sampled solution from Θ, sampled according to distribution r. Notice that r induces a distribution on the random variable $g(\mathbf{X})$, the value of the objective function at $\mathbf{X}$, making quantities such as $E_r[g(\mathbf{X})]$ or the $(1-\alpha)$-quantile of $g(\mathbf{X})$ well defined.

Hu et al. [18] show that there is a recursive sequence of reference distributions $\{r_k; k = 0, 1, 2, \ldots)$ on Θ with the property that

$$\lim_{k\to\infty} E_{r_k}[g(\mathbf{X})] = g(\mathbf{x}^*).$$

Because $\mathbf{x}^*$ is unique, this sequence of distributions converges to a distribution that places probability 1 on $\mathbf{x}^*$. Specifically, starting from some initial distribution $r_0(\mathbf{x})$ that assigns positive probability to all $\mathbf{x} \in \Theta$,

$$r_{k+1}(\mathbf{x}) = \frac{g(\mathbf{x})r_k(\mathbf{x})}{\sum_{\mathbf{z}\in\Theta} g(\mathbf{z})r_k(\mathbf{z})}.$$

In theory, if we could generate samples from r_k, we could empirically estimate r_{k+1}, and continue to do this until the reference distribution essentially degenerates onto $\mathbf{x}^*$. Unfortunately, r_k may have no special structure, making sampling from it computationally difficult. Therefore, Hu et al. [18] suggest that sampling from Θ be from a convenient parametric distribution $f(\cdot;\boldsymbol{\beta}_k)$, where at each iteration we choose $\boldsymbol{\beta}_k$ to minimize the Kullback–Leibler divergence between the parametric distribution and the reference distribution; this is what Step 3 in the MRAS algorithm below does.

MRAS

(1) Choose initial distribution $f(\cdot;\boldsymbol{\beta}_0)$ over Θ, $\alpha \in (0, 1]$ and $k = 0$.

(2) Calculate γ_{k+1}, the $(1-\alpha)$-quantile of $g(\mathbf{X})$ with $\mathbf{X} \sim f(\cdot;\boldsymbol{\beta}_k)$.

(3) Solve for

$$\boldsymbol{\beta}_{k+1} = \arg\max_{\boldsymbol{\beta}} E_{\beta_k}\left\{\frac{[g(\mathbf{X})]^k}{f(\mathbf{X},\boldsymbol{\beta}_k)}\mathcal{I}(g(\mathbf{X}) \geq \gamma_{k+1}) \ln f(\mathbf{X},\boldsymbol{\beta})\right\}.$$

(4) Let $k = k + 1$; go to 2.

The MRAS algorithm above assumes that the $(1-\alpha)$-quantile in Step 2 and the expectation in Step 3 can be calculated exactly. In practice they are estimated via samples from Θ according to $f(\cdot, \boldsymbol{\beta}_k)$. The quantile threshold in Step 2 selects out only the most promising samples in Step 3 to use for updating $\boldsymbol{\beta}$; convergence of the MRAS algorithm does not depend on the choice of α, but empirical performance does. When the simulation output is stochastic, then $g(\mathbf{x})$ also must be estimated, as described in Hu et al. [17].

MRAS avoids the need to make a decision between searching globally and locally.

4.3. Locally Convergent ARS

All of the random search algorithms that we have discussed so far are globally convergent, which means that the search will eventually find a globally optimal solution of the DOvS problem as the simulation effort goes to infinity. Unless we assume that the DOvS problem has special structure, the only way to ensure global convergence is to evaluate all feasible solutions with an infinite number of replications. Globally convergent ARS algorithms are designed to make sure that all solutions are simulated infinitely often in the limit, while aggressively pursuing good solutions subject to that requirement.

If we could actually evaluate all feasible solutions, then we could treat the DOvS problem as an R&S problem. In practice, however, the total number of feasible solutions can be very large, and only a small fraction of them can be simulated. In that situation, global convergence has little practical meaning.

A different approach is to seek a convergence property that can be used to design a practically meaningful stopping criterion for the algorithm. An example is the local convergence criterion initiated by Hong and Nelson [14]. Let $\mathcal{N}(\mathbf{x}) = \{\mathbf{y}\colon \mathbf{y} \in \Theta \text{ and } \|\mathbf{x} - \mathbf{y}\| = 1\}$ be the local neighborhood of $\mathbf{x} \in \Theta$, where $\|\mathbf{x} - \mathbf{y}\|$ denotes the Euclidean distance between $\mathbf{x}$ and $\mathbf{y}$. Hong and Nelson [14] define $\mathbf{x}$ as a *local minimizer* if $\mathbf{x} \in \Theta$ and either $\mathcal{N}(\mathbf{x}) = \varnothing$ or $g(\mathbf{x}) \le g(\mathbf{y})$ for all $\mathbf{y} \in \mathcal{N}(\mathbf{x})$.

Notice that, to statistically establish the global optimality of a solution in DOvS, all solutions in Θ have to be simulated (unless there is special structure). To check the local optimality of a solution, however, only the solutions in its local neighborhood need to be simulated and the local neighborhood has at most $2d$ solutions. Therefore, it is more practical to verify local optimality than global optimality. Xu et al. [34] showed that an R&S procedure due to Kim [19] can be applied to test the local optimality of a solution with controlled Types I and II errors by sequentially simulating all solutions in the local neighborhood of a candidate optimal along with the candidate itself. Thus, local optimality can be established with statistical confidence while simulating only a small fraction of the solutions in Θ.

Hong and Nelson [15] designed a framework for locally convergent DOvS algorithms and showed that local convergence can be achieved while still leaving a great deal of freedom to design efficient and aggressive DOvS algorithms. The COMPASS algorithm of Hong and Nelson [14] is an instance of this framework. The following is a high-level description of COMPASS.

COMPASS

(1) Either randomly or by design, select some initial feasible solutions and simulate them.

(2) Build the most promising area around the current sample best solution based on geometry.

(3) Sample new solutions from the most promising area.

(4) Simulate the new solutions and the solutions that define the most promising area a little bit more.

(5) Calculate the cumulative sample average for each active solution, and choose the solution with the best cumulative sample average as $\mathbf{x}_k^*$. Go to Step 2.

The "most promising area" is the part of Θ that we want to explore most intensely. In the original COMPASS algorithm, the most promising area consisted of all feasible solutions that have not yet been simulated, and are closer to the current sample best solution than to any other feasible solution that has been simulated. Specifically,

$$\{\mathbf{x}\colon \mathbf{x}\in\Theta \text{ and } \|\mathbf{x}-\hat{\mathbf{x}}_k^*\|\le\|\mathbf{x}-\mathbf{y}\|,\ \forall \mathbf{y}\in\mathcal{V}_k \text{ and } \mathbf{y}\ne\hat{\mathbf{x}}_k^*\},$$

where $\mathcal{V}_k$ is the set of all visited solutions through iteration k. This area is easily defined by linear constraints, but other choices are possible (Xu et al. [34]). By continuing to improve the estimates of $g(\mathbf{x})$ for the "active" solutions—solutions in the most promising area and those that define it—the algorithm has a way to back out from the current most promising area if no locally optimal solution is discovered.

5. Stochastic Constraints

For many problems it is natural to have a stochastic constraint $E[C(\mathbf{x})]\le q$. Notice that a chance constraint is of this form when $C(\mathbf{x})$ is an indicator random variable that takes the value 1 if the event of interest happens, 0 otherwise.

In principle, we could take DOvS algorithms like BEESE or COMPASS and roll the constraint into the output estimator as a penalty, e.g.,

$$\bar{Y}(\mathbf{x};n)+b(\mathbf{x};n)\cdot[\bar{C}(\mathbf{x};n)-q]^+.$$

Another approach is to first run simulations at a carefully designed collection of $\mathbf{x}$'s, fit a metamodel $\hat{c}(\mathbf{x})$ that approximates $E[C(\mathbf{x})]$, and then include $\hat{c}(\mathbf{x})\le q$ as just another deterministic constraint.

Batur and Kim [5] have proposed using ideas from R&S to check the feasibility of all $\mathbf{x}\in\Theta$ with respect to a stochastic constraint. Instead of classifying each solution as either feasible or infeasible with respect to $E[C(\mathbf{x})]\le q$, they recognize that in a stochastic setting it may be reasonable to allow some relaxation of the constraint. In particular, they suggest the user provide an $\varepsilon>0$ such that they are comfortable with the following.

- If $E[C(\mathbf{x})]\le q-\varepsilon$, then it is *desirable* to declare that $\mathbf{x}$ is feasible: $\mathbf{x}\in D$.
- If $q-\varepsilon<E[C(\mathbf{x})]\le q+\varepsilon$, then it is *acceptable* to declare that $\mathbf{x}$ is feasible or infeasible: $\mathbf{x}\in A$.
- If $E[C(\mathbf{x})]>q+\varepsilon$, then it is *unacceptable* to declare that $\mathbf{x}$ is feasible: $\mathbf{x}\in U$.

Batur and Kim [5] provide a sequential classification procedure that returns a subset $\mathcal{F}\subset\Theta$ with the guarantee that $\Pr\{D\subset\mathcal{F}\subset(D\cup A)\}\ge 1-\alpha$. The procedure is valid as stated for normally distributed $C(\mathbf{x})$ and valid as $\varepsilon\to 0$ otherwise. Batur and Kim [5] show that the procedure can be effective for $|\Theta|\approx 1{,}000$.

But what if $\mathbf{x}$ has to satisfy $s>1$ stochastic constraints that may be correlated? Then, of course, all constraints must be checked, and significantly more simulation may be required to provide the probability guarantee simultaneously for all of them. To speed the decision that a solution $\mathbf{x}$ is unacceptable, Batur and Kim [5] also check the linear combination constraint

$$\sum_{i=1}^{s}a_iE[C_i(\mathbf{x})]\le\sum_{i=1}^{s}a_iq_i$$

for constants $\mathbf{a}=(a_1,a_2,\ldots,a_s)$. Notice that any unacceptable $\mathbf{x}$ is also unacceptable for this constraint; therefore, adding this constraint can accelerate recognition of an unacceptable

solution. The choice of the constant vector **a** matters, and because the optimal **a** is typically unknown in advance, Batur and Kim [5] propose a heuristic.

6. Solvers

In this final section we discuss the use of DOvS solvers.

6.1. Industrial Strength COMPASS

To further improve the finite-time performance of the COMPASS algorithm and make it competitive with commercial DOvS solvers, Xu et al. [34] developed the Industrial Strength COMPASS (ISC). The ISC has the COMPASS algorithm as its core but adds other steps to make the algorithm more efficient in solving practical problems.

The ISC algorithm divides the optimization process into three phases: a global phase that explores the entire feasible region and identifies several areas that possibly contain competitive locally optimal solutions, a local phase that searches for a locally optimal solution in each of the regions identified in the global phase, and a cleanup phase that selects the best solution among all solutions identified as locally optimal and estimates its true expected performance to within a user-provided precision.

The ISC uses a niching genetic algorithm for the global phase, the COMPASS algorithm for the local phase, and an R&S procedure for the cleanup phase. It also defines meaningful and testable transition rules between the phases. Xu et al. [34] compared the ISC with a stand-alone version of OptQuest—a commercial solver that is embedded in many commercial simulation packages—for a number of examples with different characteristics. They found that the ISC is competitive with OptQuest, despite enforcing finite-sample and asymptotic guarantees. The `C++` implementation of ISC with its instructions and related papers are free for downloading from http://www.iscompass.net.

6.2. Using Commercial OvS Software

This section is based on Hong and Nelson [16] and Chapter 12 of Banks et al. [3].

As noted earlier, a standard feature of simulation modeling software is an OvS tool. Considering the difficulty of establishing connections between a stand-alone OvS solver and simulation modeling software, it is not surprising that most simulation practitioners use the integrated OvS tool. In this section we provide some guidance on using them to obtain good results.

Robust heuristics are the most common algorithms found in the integrated OvS tools. A "robust heuristic" is an OvS procedure that does not depend on strong problem structure to be effective, and is somewhat tolerant of some sampling variability. Genetic algorithms and tabu search are two prominent examples, but there are others and many variations. Most of the integrated tools can be applied to problems with continuous, integer, and categorical decision variables. These heuristics have been observed to be highly effective on difficult *deterministic* optimization problems, but they usually provide no performance guarantees, either in the deterministic or stochastic case. Therefore, we offer three suggestions to make them more effective for OvS: control sampling variability, restart, and cleanup.

6.2.1. Control Sampling Variability. In many cases, it will be up to the simulation user to decide how many replications will be obtained at each feasible solution examined by the solver. This is a difficult problem in general. Ideally, the number of replications should increase as the heuristic discovers better and better solutions because it is statistically much more difficult to distinguish solutions that are close in performance than ones that differ substantially. At the beginning of the search, very little error control may be needed for the heuristic to identify good solutions and search directions; later in the search, however, this might not be the case.

If the OvS software has an "adaptive" setting—meaning it adjusts the number of replications based on the variance of the simulation estimates—then use it. If, on the other hand, you must specify a fixed number of replications per solution, then a preliminary experiment should be conducted: Simulate several solutions, some at the extremes of the feasible region and some nearer the center. Compare the apparent best and apparent worst of these solutions. Find the minimum number of replications required to declare them to be statistically significantly different. This is the minimum number of replications that should be used, which may be substantially more than the default minimum number of replications specified by the OvS tool because the software designers want their tool to deliver results quickly. But remember, if the decision that the optimization will support is important, then waiting hours or even days for the *best* decision may well be worth the wait.

6.2.2. Restarting the Optimization. Because robust heuristics provide no guarantees that they converge to the optimal solution, it makes sense to run the optimizer several times to see which run yields the best solution. Each optimization run should use different random number seeds or streams and should start from different initial solutions. Be sure to start at solutions on the extremes of the solution space, in the center of the space, or even at randomly generated solutions. If people familiar with the system suspect that certain solutions will be good, include them as starting solutions too. The inconvenience of initializing several optimization runs is worth it if it leads to a better solution.

6.2.3. Statistical Cleanup. After the optimization run or runs have completed, it is critical to perform a second set of experiments on the top solutions identified by the heuristic; we call these "cleanup experiments." In an OvS problem you can never be sure you have found the optimal solution (or that you even simulated it, unless you exhausted Θ); that is an error that you have to live with. But there are two additional types of error that are avoidable: failing to recognize the best solution that actually *was* simulated during the search, and poorly estimating the performance of the solution that is selected in the end. These errors occur because no optimization algorithm can hope to make any progress while at the same time maintaining statistical error control every step of the way, and there is a natural bias toward solutions that, by chance, received favorable simulation estimates. Therefore, we recommend performing a rigorous statistical analysis employing the comparison techniques described in §3 to decide which are the best or near best solutions discovered during the search. Include the apparent top 5% to 10% of the solutions simulated during the search in this controlled experiment. The "cleanup" concept was introduced in Boesel et al. [7].

In summary, the outcomes from using commercial OvS software can be improved by (a) doing a bit of preliminary study to assess output variability, (b) making a number of optimization runs to improve the chances of identifying good solutions, and (c) performing a sound statistical analysis of the top solutions to provide a statistical guarantee of selecting the best among them and estimating its performance well.

Acknowledgments

The author acknowledges prior publication of portions of this paper in Hong and Nelson [16].

References

[1] S. Andradóttir. Accelerating the convergence of random search methods for discrete stochastic optimization. *ACM Transactions on Modeling and Computer Simulation* 9:349–380, 1999.

[2] S. Andradóttir. An overview of simulation optimization via random search. S. Henderson and B. L. Nelson, eds. Chapter 20 in *Handbooks in Operations Research and Management Science: Simulation*, Chapter 20. North-Holland, Amsterdam, 2006.

[3] J. Banks, J. S. Carson, B. L. Nelson, and D. Nicol. *Discrete-Event System Simulation*. 5th ed. Prentice Hall, Upper Saddle River, NJ, 617–631, 2010.

[4] R. R. Barton and M. Meckesheimer. Metamodel-based simulation optimization. S. Henderson and B. L. Nelson, eds. *Handbooks in Operations Research and Management Science: Simulation*, Chapter 18. North-Holland, Amsterdam, 535–574, 2006.

[5] D. Batur and S.-H. Kim. Finding feasible systems in the presence of constrains on multiple performance measures. *ACM Transactions on Modeling and Computer Simulation*. Forthcoming.

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

[7] J. Boesel, B. L. Nelson, and S.-H. Kim. Using ranking and selection to "clean up" after simulation optimization. *Operations Research* 51:814–825, 2003.

[8] J. Branke, S. E. Chick, and C. Schmidt. Selecting a selection procedure. *Management Science* 53:1916–1932, 2007.

[9] J. A. Buzacott and J. G. Shantikumar. *Stochastic Models of Manufacturing Systems*. Prentice-Hall, Englewood Cliffs, NJ, 1993.

[10] M. C. Fu. Optimization for simulation: Theory vs. practice. *INFORMS Journal on Computing* 14:192–215, 2002.

[11] M. C. Fu. What you should know about simulation and derivatives. *Naval Research Logistics* 55:723–736, 2008.

[12] D. Goldsman and B. L. Nelson. Comparing systems via simulation. J. Banks, ed. *Handbook of Simulation*. John Wiley & Sons, New York, 273–306, 1998.

[13] W. J. Gutjahr, A. Hellmayr, and G. Ch. Pflug. Optimal stochastic single- machine-tardiness scheduling by stochastic branch-and-bound. *European Journal of Operations Research* 117:396–413, 1999.

[14] L. J. Hong and B. L. Nelson. Discrete optimization via simulation using COMPASS. *Operations Research* 54:115–129, 2006.

[15] L. J. Hong and B. L. Nelson. A framework for locally convergent random search algorithms for discrete optimization via simulation. *ACM Transactions on Modeling and Computer Simulation* 17:19/1–19/22, 2007.

[16] L. J. Hong and B. L. Nelson. A brief introduction to optimization via simulation. M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin, and R. G. Ingalls, eds. *Proceedings of the 2009 Winter Simulation Conference*, Institute of Electrical and Electronics Engineers, Piscataway, NJ, 75–85, 2009.

[17] J. Hu, M. C. Fu, and S. I. Marcus. Stochastic optimization using model reference adaptive search. M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, eds. *Proceedings of the 2005 Winter Simulation Conference*, Institute of Electrical and Electronics Engineers, Piscataway, NJ, 811–818, 2005.

[18] J. Hu, M. C. Fu, and S. I. Marcus. A model reference adaptive search method for global optimization. *Operations Research* 55:549–568, 2007.

[19] S.-H. Kim. Comparison with a standard via fully sequential procedure. *ACM Transactions on Modeling and Computer Simulation* 15:1–20, 2005.

[20] S. Mahajan and G. Van Ryzin. Stocking retail assortments under dynamic consumer substitution. *Operations Research* 49:334–351, 2001.

[21] H. Mühlenbein and G. Paaß. From recombination of genes to the estimation of distributions: I. Binary parameters. H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, eds. *Parallel Problem Solving in Nature–PPSN IV*. Springer-Verlag, Berlin, 178–197, 1996.

[22] B. L. Nelson and D. Goldsman. Comparisons with a standard in simulation experiments. *Management Science* 47:449–463, 2001.

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

[24] V. I. Norkin, Y. M. Ermoliev, and A. Ruszczyński. On optimal allocation of indivisibles under uncertainty. *Operations Research* 46:381–395, 1998.

[25] V. I. Norkin, G. Ch. Pflug, and A. Ruszczyński. A branch and bound method for stochastic global optimization. *Mathematical Programming* 83:425–450, 1998.

[26] S. Ólafsson. Metaheuristics. S. Henderson and B. L. Nelson, eds. Chapter 21 in *Handbooks in Operations Research and Management Science: Simulation.* North-Holland, Amsterdam, 633–654, 2006.

[27] J. Pichitlamken and B. L. Nelson. A combined procedure for optimization via simulation. *ACM Transactions on Modeling and Computer Simulation* 13:155–179, 2003.

[28] J. Pichitlamken, B. L. Nelson, and L. J. Hong. A sequential procedure for neighborhood selection-of-the-best in optimization via simulation. *European Journal of Operational Research* 173:283–298, 2006.

[29] A. A. Prudius and S. Andradóttir. Balanced explorative and exploitative search with estimation for simulation optimization. *INFORMS Journal on Computing* 21:193–208, 2009.

[30] R. Y. Rubinstein and D. P. Kroese. *The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte Carlo Simulation, and Machine Learning.* Springer, New York, 2004.

[31] L. Shi and S. Ólafsson. Nested partitions method for stochastic optimization. *Methodology and Computing in Applied Probability* 2:271–291, 2000.

[32] L. Shi and S. Ólafsson. *Nested Partitions Method, Theory and Applications.* Springer, New York, 2009.

[33] R. R. Wilcox. A table for Rinott's selection procedure. *Journal of Quality Technology* 16:97–100, 1984.

[34] J. Xu, L. J. Hong, and B. L. Nelson. Industrial strength COMPASS: A comprehensive algorithm and software for optimization via simulation. *ACM Transactions on Modeling and Computer Simulation* 20:1–29, 2010.

[35] W. L. Xu. Flexibility, lifecycle planning and simulation-based optimization in integrated supply chains. Ph.D. thesis, Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL, 2010.

[36] D. Yan and H. Mukai. Stochastic discrete optimization. *SIAM Journal of Control and Optimization* 30:594–612, 1992.