Chapter 2
Discrete Optimization via Simulation

L. Jeff Hong, Barry L. Nelson, and Jie Xu

Abstract This chapter describes tools and techniques that are useful for optimization via simulation—maximizing or minimizing the expected value of a performance measure of a stochastic simulation—when the decision variables are discrete. Ranking and selection, globally and locally convergent random search and ordinal optimization are covered, along with a collection of "enhancements" that may be applied to many different discrete optimization via simulation algorithms. We also provide strategies for using commercial solvers.

2.1 Introduction

In this chapter we cover optimization via simulation (OvS) problems that can be represented as

$$\min g(\mathbf{x}), \quad \mathbf{x} \in \Theta, \tag{2.1}$$

where $g(\mathbf{x}) = \mathsf{E}[Y(\mathbf{x}, \xi)]$. There is a single objective $g(\mathbf{x})$, which is representable as the expected value of a random variable $Y(\mathbf{x}, \xi)$, where ξ represents the randomness, e.g., the random numbers in a simulation. The distribution of $Y(\mathbf{x}, \xi)$ is an unknown function of the vector of decision variables $\mathbf{x}$, but realizations of $Y(\mathbf{x}, \xi)$ can be observed through simulation experiments. If the problem is more naturally thought of as maximization then we can always formulate an equivalent minimization version. Henceforth we drop the argument ξ for notational convenience.

L.J. Hong
City University of Hong Kong, Kowloon Tong, Hong Kong
e-mail: jeffhong@cityu.edu.hk

B.L. Nelson
Northwestern University, Evanston, IL, USA
e-mail: nelsonb@northwestern.edu

J. Xu (✉)
George Mason University, Fairfax, VA, USA
e-mail: jxu13@gmu.edu

M.C. Fu (ed.), *Handbook of Simulation Optimization*, International Series in Operations Research & Management Science 216, DOI 10.1007/978-1-4939-1384-8_2

Our focus is on two related cases: Either $\mathbf{x}$ is an index in the set $\Theta = \{1, 2, \ldots, k\}$ of feasible solutions, which may be categorical (not ordered in any way); or $\mathbf{x}$ is a vector of d integer-ordered decision variables in a feasible region $\Phi \subset \mathbb{R}^d$, possibly defined by a set of deterministic constraints. In this case we assume that Φ is compact and convex. Therefore, $\Theta = \Phi \cap \mathbb{Z}^d$ is a finite set, where $\mathbb{Z}^d$ denotes all d-dimensional vectors with integer components, and Problem (2.1) has only a finite number of feasible solutions. We refer to this class of problems as Discrete Optimization via Simulation (DOvS) problems.

The solution methods we describe assume that $\mathrm{Var}[Y(\mathbf{x})] < \infty$ for all $\mathbf{x} \in \Theta$, and that we have an estimator $\hat{g}(\mathbf{x})$ that converges with probability 1 (w.p.1) to $g(\mathbf{x})$ as we expend more and more simulation effort on solution $\mathbf{x}$. If that estimator is a sample mean, then we use the notation $\overline{Y}(\mathbf{x})$. Often, but not always, we can simulate independent and identically distributed (i.i.d.) replications, $Y_1(\mathbf{x}), Y_2(\mathbf{x}), \ldots$ at any $\mathbf{x}$.

The following examples, which are based on Nelson [48], illustrate the types of problems we consider and the issues that arise.

2.1.1 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.*

In this problem there are a small number of feasible solutions, and they can be treated as categorical. This is a choice, however, because we could also define $\mathbf{x} = (x_1, x_2, \ldots, x_d)$ where x_i is the number of redundant components of type i to include. The state of art for solving DOvS problems makes it advantageous to treat the problem as categorical if the number of solutions is relatively small because there are highly efficient solution methods that apply when it is possible to simulate all feasible solutions, at least a little. This is analogous to deterministic integer programming (IP) problems in which it is possible to exhaust the feasible region, making it pointless to apply a high-powered IP algorithm. However, unlike a deterministic IP, a single evaluation of the objective function in DOvS is (typically) not sufficient because $Y(\mathbf{x})$ is only an estimator of $g(\mathbf{x})$ that is subject to sampling error, i.e., $Y(x) \neq g(\mathbf{x})$ w.p.1, even though $\mathsf{E}[Y(x)] = g(\mathbf{x})$. Sampling error is reduced by expending additional simulation effort at solution $\mathbf{x}$, and doing so (usually adaptively) is the central feature of solution methods.

In this problem $g(\mathbf{x})$ represents steady-state (long-run average as time goes to infinity) system unavailability, which implies that the estimator of $g(\mathbf{x})$ can be defined in one of two ways:

Extended replication average:

$$\hat{Y}(\mathbf{x}) = \frac{1}{T}\int_0^T A(t)\,dt \tag{2.2}$$

where $A(t) = 1$ if the system is unavailable, and 0 otherwise. Sampling error is controlled by increasing the run length T.

Additional replication average:

$$\overline{Y}(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{1}{T_e - T_b}\int_{T_b}^{T_e} A_i(t)\,dt\right) = \frac{1}{n}\sum_{i=1}^{n}\hat{Y}_i(\mathbf{x}) \tag{2.3}$$

where $T_e > T_b$ are fixed times, and $A_i(t)$ is the sample path from the ith replication. Sampling error is controlled by increasing the number of i.i.d. replications n.

In Sect. 2.3 we describe ranking and selection methods for such problems.

2.1.2 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 1-year planning horizon is maximized. The deterministic constraints are $x_1+x_2+x_3 \le 20, x_4+x_5 = 20, 1 \le x_i \le 20$ and $x_i \in \mathbb{Z}^+$ for $i = 1,2,\ldots,5$, implying $21,660$ feasible solutions. This example is adapted from [8, 58] .

Here the number of feasible solutions is probably larger than can be exhausted, so some sort of search is required. In Sects. 2.5–2.6 we describe methods to solve such problems based on adaptive random search.

Notice in this problem that the fixed 1-year planning horizon means that sampling error is reduced only by increasing the number of replications—it makes no sense to extend a replication beyond 1 year because the performance measure of interest is defined with respect to 1 year. Thus, the natural estimator is

$$\overline{Y}(\mathbf{x}) = \frac{1}{n}\sum_{j=1}^{n} Y_j(\mathbf{x})$$

where $Y_j(\mathbf{x})$ is the throughput observed in 1 year of production on replication j for solution $\mathbf{x}$.

2.1.3 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. This example is adapted from [45].*

In theory, there are a countably infinite number of feasible solutions since there is no fixed upper bound on the quantity of variant i that the retailer stocks, x_i. In practice, clearly there is a level beyond which it makes no sense to stock. These level bounds can be determined by factors such as store capacity and maximal demand possible. Then, the problem can be transformed into a DOvS problem.

When solving deterministic IPs, branch-and-bound methods that relax integrality constraints are often used, which is possible because the objective function can be evaluated at non-integer 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 stocking 112.3 blue shirts. 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. And, of course, $g(\mathbf{x})$ can only be estimated.

2.1.4 Themes

The focus of this chapter is on methods that lead to rigorously provable performance, as defined in Sect. 2.2 below (however, in Sect. 2.8 we give practically useful tips for using commercial OvS solvers based on metaheuristics). We assume little or no structural information about $g(\mathbf{x})$; as a result, the balance between expending effort on search—looking for better feasible solutions—and estimation—estimating the true objective function value of solutions we have investigated—is a core issue for DOvS algorithms.

There are three fundamental types of errors that occur in DOvS problems; the latter two occur even when the number of feasible solutions is small enough that all of them can be simulated.

1. *The optimal solution is never simulated.* This is a reality in many difficult nonlinear IPs when the feasible solutions cannot be exhausted, and DOvS is no different.

2. *The best solution that was simulated is not selected.* Sampling variability means that the best solution we simulated may not have the best estimated objective function value.
3. *We do not have a good estimate of the objective function value of the solution we do select.* When minimizing a stochastic response by selecting the solution with the smallest estimated value, there is a natural bias toward solutions whose estimated performance is better (lower) than its true expected value.

How these issues are addressed in different problem classes is the subject of the remainder of this chapter.

2.2 Optimality Conditions

Let $\Theta^* = \text{argmin}\{g(\mathbf{x}) : \mathbf{x} \in \Theta\}$ be the set of optimal solutions of Problem (2.1). Because $\Theta = \Phi \cap \mathbb{Z}^d$ and $\Phi \subset \mathbb{R}^d$ is a compact set, Θ is a finite set, i.e., $|\Theta| < \infty$. Therefore, Θ^* is guaranteed to be nonempty. Furthermore, the finiteness of Θ also implies that there exists a positive constant $\delta > 0$ such that

$$g^* \leq g(\mathbf{y}) - \delta \quad \text{forall } \mathbf{y} \in \Theta \setminus \Theta^*, \tag{2.4}$$

where $g^* = \min_{\mathbf{x}\in\Theta} g(\mathbf{x})$ is the optimal objective value. Note that Eq. (2.4) implies that optimal solutions are at least δ better than other feasible solutions.

To solve optimization problems we often need optimality conditions which (a) assure the correctness of algorithms and (b) help in designing implementable stopping rules. To illustrate the usefulness of optimality conditions, we consider the following two examples.

- In unconstrained nonlinear optimization, convergence to a stationary point whose gradient is zero is a widely used optimality condition. Many algorithms, including the steepest descent algorithm and Netwon's method, are proved to satisfy this condition [50]. In practice, all these algorithms typically stop short of convergence. But they often stop when they find a solution whose gradient is sufficiently close to zero.
- In integer linear programming, algorithms often keep track of an upper bound and a lower bound. A commonly used optimality condition is that the gap between the two bounds goes to zero. Many algorithms, including the branch-and-bound and branch-and-cut algorithms, are proved to satisfy this condition. Then in practical implementation, these algorithms often stop when the gap is small enough.

Although the set of optimal solutions Θ^* is clearly defined for DOvS problems, defining optimality conditions for DOvS algorithms is not easy for the following reasons:

1. The objective function $g(\mathbf{x})$ cannot be calculated exactly; instead, it can only be estimated by $\overline{Y}(\mathbf{x})$. This estimation noise generally makes it impossible to rank

solutions with 100 % confidence. Therefore, DOvS algorithms cannot guarantee in general to find an optimal solution with a finite amount of computational effort.
2. Typically $g(\mathbf{x})$ and $Y(\mathbf{x})$ are unknown functions that are embedded in simulation models. We do not have the structural results on $g(\mathbf{x})$ or $Y(\mathbf{x})$ that can be used to screen out (often a large number of) inferior solutions as, for instance, in branch-and-cut algorithms for integer linear programs. Therefore, to find an optimal solution of Problem (2.1), one has to evaluate all feasible solutions.
3. Although Θ is a finite set, it often has a large number of feasible solutions as in the flow-line and inventory-management examples reported in Sects. 2.1.2 and 2.1.3, respectively. Complete enumeration of all solutions may not be possible for many practical problems.

Despite these difficulties, researchers have established various optimality conditions for DOvS problems that are either theoretically convenient or practically useful. In this section we will introduce these conditions.

When Θ is small, i.e., Θ has less than a few hundreds of solutions, we may be able to simulate all solutions and select the best among them. This is known as ranking and selection (R&S). Because of the randomness in the simulation outputs, one cannot guarantee to select the best solution with 100 % confidence. Then, a practical approach is to analyze *the probability of correct selection* (PCS), i.e., $\mathsf{P}(\mathbf{x}^* \in \Theta^*)$ where $\mathbf{x}^*$ denotes the selected best solution. A commonly used optimality condition is to require R&S algorithms to achieve a predetermined PCS, i.e., $\mathsf{P}(\mathbf{x}^* \in \Theta^*) \geq 1-\alpha$. In Sect. 2.3 we will introduce such algorithms.

When Θ is large, simulating all solutions in Θ becomes practically impossible. One may soften the goal of finding an optimal solution to finding a good enough solution, where a "good enough" solution may be defined as one of the top t solutions in Θ. Suppose that one has the computational budget to evaluate n solutions in Θ. Then, it is important to know the probability that at least one of these n solutions is a top t solution, which is known as the *alignment probability* (AP). Let $T \subset \Theta$ denote the set of top t solutions, and let S denote the set of the chosen n solutions. Then, the alignment probability is defined as $\mathsf{P}(|T \cap S| \geq 1)$. A commonly used optimality condition is to require algorithms to achieve a predetermined level of AP, i.e., $\mathsf{P}(|T \cap S| \geq 1) \geq 1-\alpha$. This optimality condition is used in ordinal optimization, which will be introduced in Sect. 2.4.

Another commonly used optimality condition when Θ is large is global convergence as the amount of computational effort goes to infinity. Let $\mathbf{x}^*_m$ denote the solution that the algorithm would report as optimal if stopped at the end of iteration m and $g^*_m = g(\mathbf{x}^*_m)$. Then, the algorithm is globally convergent in probability if $g^*_m \to g^*$ in probability and globally convergent w.p.1 if $g^*_m \to g^*$ w.p.1. By Eq. (2.4), these two convergence criterion are equivalent to

$$\lim_{m\to\infty} \mathsf{P}(\mathbf{x}^\star_m \in \Theta^\star) = 1,$$

$$\mathsf{P}\left(\lim_{m\to\infty} \mathbf{1}\{\mathbf{x}^\star_m \in \Theta^\star\} = 1\right) = 1,$$

respectively, where $\mathbf{1}\{\cdot\}$ is the indicator function. As all algorithms stop in a finite amount of time, one may wonder why convergence properties are desirable. Andradóttir [2] answers this question:

> Although this [convergence] performance guarantee does not assure that the algorithm will return a "good" estimated optimal solution (because additional computational effort may be required), it is certainly a reassuring property to have. From a different perspective, it is worrisome to use a simulation optimization algorithm in practice that is not known to converge even if an infinite amount of computational effort is expended!

Many globally convergent algorithms have been proposed to solve DOvS problems and we will introduce them in Sect. 2.5.

As pointed earlier in this section, optimality conditions assure the correctness of algorithms and help in designing implementable stopping rules. Global convergence achieves the first goal by reassuring the algorithm eventually finds an optimal solution. However, to achieve global convergence when there is no special structure, algorithms have to evaluate all solutions in Θ in the limit, and it is not clear how to relax this requirement for some implementable stopping rules. To resolve this problem, local convergence has been proposed. Let $N(\mathbf{x}) \subset \Theta$ denote the local neighborhood of any solution $\mathbf{x} \in \Theta$, and let $\mathbf{x}$ be a locally optimal solution if $g(\mathbf{x}) \leq g(\mathbf{y})$ for all $\mathbf{y} \in N(\mathbf{x})$. Notice that the definition of local optimality depends on the definition of the local neighborhood and different local neighborhoods may result in different local optimal solutions. Let $\mathscr{L}$ denote the set of locally optimal solutions for the DOvS problem. Then, similar to the definition of global convergence, we may define local convergence in probability and local convergence w.p.1 as

$$\lim_{m\to\infty} \mathsf{P}(\mathbf{x}_m^\star \in \mathscr{L}) = 1,$$

$$\mathsf{P}\left(\lim_{m\to\infty} \mathbf{1}\{\mathbf{x}_m^\star \in \mathscr{L}\} = 1\right) = 1,$$

respectively. To converge to a locally optimal solution, algorithms do not need to evaluate all feasible solutions. Furthermore, because the local neighborhood is typically a small set, one can statistically test the local optimality of any solution $\mathbf{x}$, i.e.,

$$\mathrm{H}_0 : g(\mathbf{x}) \leq \min_{\mathbf{y}\in N(\mathbf{x})} g(\mathbf{y}) \quad \text{vs} \quad \mathrm{H}_1 : g(\mathbf{x}) > \min_{\mathbf{y}\in N(\mathbf{x})} g(\mathbf{y}),$$

and control the type I and type II errors of the test. This provides an implementable stopping rule for algorithms in a finite amount of time. In Sect. 2.6 we will introduce some locally convergent algorithms.

Other than the algorithms that are built around convergence or correct-selection guarantees, there are also many algorithms that are based on heuristics for deterministic optimization algorithms, such as genetic algorithms and tabu search. These algorithms work well for difficult deterministic integer programs, and they are somewhat tolerant of sampling variabilities. However, they typically do not satisfy any optimality conditions for DOvS problems and may be misled by sampling variabilities. These algorithms are typically used in commercial Solvers and we offer some suggestions on how to use them effectively and efficiently in Sect. 2.8.

2.3 Ranking and Selection

Methods in the category of ranking and selection (R&S) apply to problems with a relatively small number of feasible solutions, such as designing the highly reliable system described in Sect. 2.1.1. The R&S procedures that have seen broad use in optimization via simulation treat the feasible solutions as categorical, meaning that they make no attempt to exploit relationships among the solutions (other than that their similarity may make the use of common random numbers effective). The definition of "relatively small" depends to some extent on how much time it takes to simulate an alternative, since R&S procedures simulate them all, but problems with up to 1,000 feasible solutions have been solved in this way. Recently, Luo et al. [44] implemented R&S procedures on a parallel computing environment with tens to hundreds of parallel processors, and solved practical DOvS problems with more than 20,000 feasible solutions.

Our focus will be on the indifference-zone formulation of optimality, as described in Sect. 2.2, but with some discussion of a Bayesian formulation. Suppose that there are $k \geq 2$ solutions in Θ, denoted as $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k$. We let $Y_j(\mathbf{x}_i)$ denote the jth observation from simulating solution $\mathbf{x}_i$. Many R&S methods assume that $Y_j(\mathbf{x}_i) \sim \mathcal{N}(g(\mathbf{x}_i), \sigma_i^2)$, where $g(\mathbf{x}_i)$ is unknown and the σ_i^2 are typically unknown and unequal. To simplify notation, we let $g(\mathbf{x}_1) \leq g(\mathbf{x}_2) \leq \cdots \leq g(\mathbf{x}_k)$ and the goal of a R&S procedure is to select solution $\mathbf{x}_1$ whose identity is unknown. Under the indifference-zone 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 at least $\delta > 0$. If there are a set of solutions whose objective values are within δ of the best solution, then all solutions in that set are acceptable.

R&S procedures were developed in the 1950s for statistical selection problems such as choosing the best treatment for a medical condition. In such contexts small numbers of alternatives with relatively equal variances (maybe even "known" variances from similar experiments) were common. Researchers have creatively built on this foundation to address problems that are of particular importance in computer simulation: unknown and unequal variances; larger numbers of feasible solutions; induced correlation across solutions due to the use of common random numbers; autocorrelation within a replication of a solution in steady-state simulation; and non-normal output data. Despite the extensive literature and the many variations that have been proposed, the foundations for most procedures can be found in three very old procedures described below.

Bechhofer's procedure [4] is one of the earliest and simplest indifference-zone selection procedures. It assumes that $\sigma_1^2 = \sigma_2^2 = \cdots = \sigma_k^2 = \sigma^2$ and σ^2 is known, and $Y_j(\mathbf{x}_i)$ is independent of $Y_n(\mathbf{x}_m)$ whenever $i \neq m$ (different solutions) or $j \neq n$ (different observations) or both.

Bechhofer's procedure determines the sample sizes required for all k solutions based on the variance of the solutions and by assuming $g(\mathbf{x}_1) + \delta = g(\mathbf{x}_2) = \cdots = g(\mathbf{x}_k)$—the most difficult case—which frees it from needing to know anything about

the true means. This procedure can be thought of as a hypothesis test with controlled power for detecting that one solution is $\geq \delta$ better than the others; it provides an experiment design that leads to the indifference-zone definition of optimality being satisfied with prespecified probability.

Bechhofer's Procedure

Step 1. Determine the constant h that satisfies $\mathsf{P}(Z_i \leq h, i = 1,2,\ldots,k-1) = 1-\alpha$ where $(Z_1, Z_2, \ldots, Z_{k-1})$ has a multivariate normal distribution with means 0, variances 1, and common pairwise correlations $1/2$. Let

$$n = \left\lceil \frac{2h^2\sigma^2}{\delta^2} \right\rceil.$$

Step 2. Take n observations from each solution and calculate $\overline{Y}(\mathbf{x}_i;n)$ for all $i = 1,2,\ldots,k$, where $\overline{Y}(\mathbf{x}_i;n)$ denotes the sample mean of $Y_j(\mathbf{x}_i), j = 1,2,\ldots,n$.
Step 3. Select the solution with the smallest sample mean $\overline{Y}(\mathbf{x}_i;n)$ as the best.

A feature of Bechhofer's procedure and its descendants is that they do not try to exploit information provided by the sample mean of each alternative until the very end. When the samples are collected one at a time, as they are in simulation experiments, it is possible to evaluate the selection decision at intermediate stages. Fully-sequential procedures evaluate after every sample is taken. The simplest fully-sequential procedure is Paulson's procedure [54], which makes the same assumptions as Bechhofer's procedure.

Paulson's Procedure

Step 1. Let $0 < \lambda < \delta$ and

$$a = \ln\left(\frac{k-1}{\alpha}\right)\frac{\sigma^2}{\delta-\lambda}.$$

Let $I = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k\}$ and $r = 0$.
Step 2. Let $r = r+1$. Take one observation from each solution that is in I and compute $\overline{Y}(\mathbf{x}_i;r)$ for all $\mathbf{x}_i \in I$.
Step 3. Let $I^{\text{old}} = I$ and

$$I = \left\{ \mathbf{x}_i \in I^{\text{old}} : \overline{Y}(\mathbf{x}_i;r) \leq \min_{\ell \in I^{\text{old}}} \overline{Y}(\mathbf{x}_\ell;r) + \left(\frac{a}{r} - \lambda\right)^+ \right\}, \text{ where } (x)^+ \equiv \max\{0,x\}.$$

If $|I| > 1$, then go to Step 2; otherwise, select the solution in I as the best.

Paulson's procedure uses a large deviations bound to account for taking multiple looks at the data. Descendants of Paulson's procedure use bounds based on Brownian motion crossing boundaries. The efficiency that comes from eliminating noncompetitive solutions early can be substantial.

Relatively large numbers of alternatives typically means that (a) they were created by taking all feasible combinations of some more basic decision variables, and (b) many of them are not really competitive. Both Bechhofer-like and Paulson-like procedures can benefit from a prescreening step that eliminates many of the uncompetitive solutions while retaining the best with a guaranteed probability. Subset selection procedures trace back to a procedure due to Gupta [22, 23]:

Gupta's Procedure

Step 1. Determine the constant h that satisfies $\mathsf{P}(Z_i \le h, i = 1,2,\ldots,k-1) = 1-\alpha$ where $(Z_1, Z_2, \ldots, Z_{k-1})$ has a multivariate normal distribution with means 0, variances 1, and common pairwise correlations $1/2$. Select $n \ge 1$.

Step 2. Take n observations from each solution and calculate $\overline{Y}(\mathbf{x}_i; n)$ for all $i = 1,2,\ldots,n$.

Step 3. Let

$$I = \left\{ \mathbf{x}_i : \overline{Y}(\mathbf{x}_i; n) \le \min_{\ell \ne i} \overline{Y}(\mathbf{x}_\ell, n) + h\sigma\sqrt{\frac{2}{n}} \right\}.$$

Step 4. Return I.

Under the same assumptions as Bechhofer's and Paulson's procedures, Gupta's procedure guarantees that $\mathsf{P}(\mathbf{x}_1 \in I) \ge 1-\alpha$. No indifference-zone parameter is specified, and it is possible that $|I| = k$, i.e., no solution is eliminated. In practice, when k is large many solutions are screened out so that a selection procedure like Bechhofer's can be applied to a much smaller set of solutions. By appropriately spending the allowable error α between screening and selection, the desired indifference-zone optimality condition can be attained.

We now present two procedures that are based on the principles of Bechhofer, Paulson and Gupta, but have been extended to be relevant for simulation. NSGS (Nelson et al. [49]) combines subset selection like Gupta with ranking like Bechhofer. It allows unknown and unequal variances.

NSGS Procedure

Step 1. Specify a common first-stage number of replications from each solution $n_0 \ge 2$; further, set

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

the $(1-\alpha/2)^{\frac{1}{k-1}}$ 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 [72], Bechhofer et al. [5] or Goldsman and Nelson [21].

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

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

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

Step 3. Calculate the quantity

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

for all $i \neq \ell$. Form the screening subset

$$I = \{\mathbf{x}_i : \overline{Y}(\mathbf{x}_i;n_0) \leq \overline{Y}(\mathbf{x}_\ell;n_0)+W_{i\ell} \quad \text{for all } \ell \neq i\}.$$

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

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

Step 5. Take $N_i - n_0$ additional replications from all solutions $\mathbf{x}_i \in I$.

Step 6. Compute the overall sample means $\overline{Y}(\mathbf{x}_i;N_i)$ for all $\mathbf{x}_i \in I$. Select the solution $\mathbf{x}_B = \text{argmin}_{\mathbf{x}_i}\overline{Y}(\mathbf{x}_i;N_i)$ as best.

NSGS guarantees that $\mathbf{x}_B = \mathbf{x}_1$, or $g(\mathbf{x}_B)$ is within δ of $g(\mathbf{x}_1)$, and also that $g(\mathbf{x}_B) \in \overline{Y}(\mathbf{x}_B;N_B) \pm \delta$, all w.p. $\geq 1-\alpha$. 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 non-competitive ones. 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.

Procedure KN (Kim and Nelson [39]) below is a descendant of Paulson that allows unknown and unequal variances, and the use of common random numbers.

KN Procedure

Step 1. Specify common first-stage number of replications $n_0 \geq 2$. Set

$$\eta = \frac{1}{2}\left[\left(\frac{2\alpha}{k-1}\right)^{-2/(n_0-1)}-1\right].$$

Step 2. Let $I = \{\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_k\}$ be the set of solutions still in contention, and let $h^2 = 2\eta(n_0-1)$. Obtain n_0 observations $Y_j(\mathbf{x}_i), j = 1,2,\ldots,n_0$ from each solution $\mathbf{x}_i \in I$ and compute $\overline{Y}(\mathbf{x}_i;n_0)$. For all $i \neq \ell$ calculate

$$S_{i\ell}^2 = \frac{1}{n_0 - 1} \sum_{j=1}^{n_0} \left(Y_j(\mathbf{x}_i) - Y_j(\mathbf{x}_\ell) - \left[\overline{Y}(\mathbf{x}_i; n_0) - \overline{Y}(\mathbf{x}_\ell; n_0) \right] \right)^2,$$

the sample variance of the difference between solutions i and ℓ. Set $r = n_0$.

Step 3. Set $I^{\text{old}} = I$. Let

$$I = \left\{ \mathbf{x}_i : \mathbf{x}_i \in I^{\text{old}} \text{ and } \overline{Y}(\mathbf{x}_i; r) \le \overline{Y}(\mathbf{x}_\ell; r) + W_{i\ell}(r), \forall \ell \in I^{\text{old}}, \ell \ne i \right\},$$

where

$$W_{i\ell}(r) = \frac{\delta}{2r} \left(\frac{h^2 S_{i\ell}^2}{\delta^2} - r \right)^+.$$

Step 4. If $|I| = 1$, then stop and select the solution whose index is in I as the best. Otherwise, take one additional observation $Y_{r+1}(\mathbf{x}_i)$ from each solution $\mathbf{x}_i \in I$, set $r = r + 1$ and go to Step 3.

Many extensions and variations of KN have appeared in the literature. Kim and Nelson [41] proposed KN++, which is asymptotically valid even when observations are non-normal and dependent. A drawback of a fully sequential procedure, such as KN, relative to a two-stage procedure, like NSGS, is that fully sequential procedures frequently switch among the simulations of different solutions. Switching can be computationally much more costly than running simulation experiments, depending on the computing environment, offsetting the efficiency gain of the fully sequential procedures compared to two-stage procedures that require a minimum number of switches. Hong and Nelson [27] and Osogami [53] designed sequential procedures that reduce the number of switches dramatically while still maintaining the benefit of being sequential.

There are two basic paradigms for solving the selection-of-the-best problem: frequentist (described above) and Bayesian. A comprehensive reference that covers the basic theory upon which frequentist R&S procedures are based is Kim and Nelson [40]. We give a brief overview of the Bayesian approach below, based largely on Frazier and Powell [17]. For more comprehensive treatments, see [11, 16].

A Bayesian procedure consists of a sequence of decisions; the decisions include which solution $\mathbf{x}$ to simulate next, and possibly whether or not to stop the procedure and select a solution. Let $\mathbf{x}^{(j)}, j = 0, 1, 2, \ldots$ be the jth decision of which solution to simulate, which leads to obtaining the $(j+1)$st simulation observation $Y_{j+1}(\mathbf{x}^{(j)})$. The linking of the jth decision to the $(j+1)$st observation emphasizes that in a Bayesian framework we may have informative prior distributions on the values of $g(\mathbf{x}_1), g(\mathbf{x}_2), \ldots, g(\mathbf{x}_k)$ that could be used to make an intelligent decision $\mathbf{x}^{(0)}$ even when no data have yet been obtained.

Let $\mathscr{H}_j = \{\mathbf{x}^{(0)}, Y_1(\mathbf{x}^{(0)}), \mathbf{x}^{(1)}, Y_2(\mathbf{x}^{(1)}) \ldots \mathbf{x}^{(j-1)}, Y_j(\mathbf{x}^{(j-1)})\}$ denote the history up through decision $j-1$ (observation j), with $\mathscr{H}_0 = \emptyset$. The procedure is controlled by a policy π and a stopping time τ where $\tau(\mathscr{H}_j)$ yields a binary decision to stop the procedure or to apply

$$\mathbf{x}^{(j)} = \pi(\mathscr{H}_j)$$

to decide which solution $\mathbf{x}^{(j)}$ to simulate next.

The key to seeking optimal policies in the Bayesian formulation is that uncertainty about $g(\mathbf{x})$ is represented as a prior probability distribution on its value, which is updated to a posterior distribution using Bayes rule as observations are obtained. A typical choice of prior distribution is a non-informative normal-gamma prior ($g(\mathbf{x})$ is normally distributed given its posterior variance $\sigma^2(\mathbf{x})$, and $1/\sigma^2(\mathbf{x})$ has a gamma distribution); see [17] or [15]. This choice of prior leads to a generic Bayes procedure of the following form:

Procedure Generic Bayes

Step 1. Set $n(\mathbf{x}) = 0$, $\mathscr{H}_0 = \emptyset$ and $\overline{Y}(\mathbf{x}) =$ null for all $\mathbf{x} \in \Theta$, and $j = 0$.
Step 2. Let $\mathbf{x}^{(j)} = \pi(\mathscr{H}_j)$.
Step 3. Obtain observation $Y_{j+1}(\mathbf{x}^{(j)})$ and update

$$n(\mathbf{x}^{(j)}) = n(\mathbf{x}^{(j)}) + 1$$
$$\overline{Y}(\mathbf{x}^{(j)}) = \frac{1}{n(\mathbf{x}^{(j)})} \sum_{i:\mathbf{x}^{(i)}=\mathbf{x}^{(j)}} Y_{i+1}(\mathbf{x}^{(i)}).$$

Step 3. If $\tau(\mathscr{H}_{j+1})$ indicates time to stop, then return $\mathbf{x}_B = \text{argmin}_{\mathbf{x}\in\Theta}\overline{Y}(\mathbf{x})$.
Else $j = j+1$ and go to Step 2.

Step 2 as stated above masks what is really happening: The decision $\mathbf{x}^{(j)}$ is actually a function of the posterior distributions as calculated from $\mathscr{H}_j$, and these posterior updates may be easy (in the case of a conjugate prior) or numerically challenging to obtain [11, 16].

The best policy π and stopping time τ depend on the objective. For instance, a natural objective is

$$\inf_{\pi} \mathsf{E}^{\pi}\left[g(\mathbf{x}_N^{\star})\right] = \inf_{\pi} \mathsf{E}^{\pi}\left[\min_{\mathbf{x}\in\Theta} \overline{Y}_N(\mathbf{x})\right] \tag{2.5}$$

where N is a fixed simulation budget. This is equivalent to minimizing the expected opportunity cost for the selected solution within a given simulation budget. Another objective is

$$\inf_{\pi,\tau} \mathsf{E}^{\pi}\left[g(\mathbf{x}_\tau^{\star}) - c(\tau)\right] = \inf_{\pi,\tau} \mathsf{E}^{\pi}\left[\min_{\mathbf{x}\in\Theta} \overline{Y}_\tau(\mathbf{x}) + c(\tau)\right] \tag{2.6}$$

where $c(j)$ is the cost of running j simulations; this policy and stopping rule balance selection loss with the cost of additional simulation. The expectations in (2.5)–(2.6) are with respect to the posterior distributions of $g(\mathbf{x})$.

The optimal policies for (2.5) and (2.6) are the solutions to dynamic programming problems. Unfortunately, it is computationally difficult or impossible to actually achieve the optimal policy; therefore, research has focused on heuristics that are implementable and effective. These include the optimal computing budget allocation (OCBA) procedures [9], the expected value of information (EVI) procedures [12], and the knowledge gradient (KG) methods [17, 18], which are the topics of Chap. 3.

An advantage of the Bayesian formulation is its flexibility; many kinds of information or knowledge about the problem can be incorporated into the prior beliefs, leading to substantial gains in efficiency. For instance, the knowledge that two similar solutions ($\mathbf{x}_1 \approx \mathbf{x}_2$) will probably have similar values ($g(\mathbf{x}_1) \approx g(\mathbf{x}_2)$) can be exploited (e.g., [19]).

Branke et al. [7] conducted a comprehensive set of experiments to compare the performance of different R&S procedures on thousands of combinations of problem structures. They found that no R&S procedure can dominate in all situations. They also found that the Bayesian procedures are often more efficient in terms of the total number of samples required to make a decision. However, they do not provide the type of correct-selection optimality guarantee that the frequentist procedures provide.

2.4 Ordinal Optimization

Ordinal optimization (OO), introduced by Ho et al. [25] and treated in detail in the book by Ho et al. [26], proposes "soft optimization" for OvS problems when the number of feasible solutions $k = |\Theta|$ is too large for R&S methods. Ordinal optimization selects a subset S from Θ and limits further analysis to S. If we define a set T of good enough solutions in Θ, which are often the top t solutions in Θ, we are interested in the probability that at least l solutions in T are in S, i.e., $\mathrm{P}(|T \cap S| \geq l)$. This probability is referred to as the *alignment probability* (AP) and l is called the alignment level.

There are two basic ideas behind ordinal optimization:

1. Estimating the order among solutions is much easier than estimating the absolute objective values of each solution.
2. Softening the optimization goal and accepting good enough solutions leads to an exponential reduction in computational burden.

To understand the first idea, recall that estimating $g(\mathbf{x})$ with $\overline{Y}(\mathbf{x})$ only has a convergence rate of $1/\sqrt{n}$ according to the Central Limit Theorem. By comparison, if one is only interested in identifying the set of optimal solutions Θ^*, one can

achieve exponential convergence rate with respect to order using results from large deviation theory. Specifically, if $Y(\mathbf{x})$ has a finite moment generating function $M(\lambda) = \mathsf{E}[e^{\lambda Y(\mathbf{x})}]$, then for any positive constant $\delta > 0$, there exists a positive constant β such that

$$\mathsf{P}(|\overline{Y}(\mathbf{x}) - g(\mathbf{x})| > \delta) \leq e^{-n\beta}. \tag{2.7}$$

Based on this result, for any alignment level l, the misalignment probability decays exponentially, as shown in [13, 14, 26, 68]. Without loss of generality, we assume that $g(\mathbf{x}_1) < g(\mathbf{x}_2) < \ldots < g(\mathbf{x}_k)$. For simplicity, we assume that all solutions receive the same number of simulation replications n. Let $\Delta = \min_{i=1,2,\ldots,k-1}(g(\mathbf{x}_{i+1}) - g(\mathbf{x}_i))$, and $\delta = \Delta/2$. Clearly, if no sample mean $\overline{Y}(\mathbf{x}_i)$ deviates from its true mean $g(\mathbf{x}_i)$ by more than δ, then all sample means are in the same order as the true means, and thus all solutions are aligned. Therefore, for misalignment to happen, there must exist some $\mathbf{x}_i$ such that $|\overline{Y}(\mathbf{x}_i) - g(\mathbf{x}_i)| \geq \delta$. We thus have the following inequality to bound the misalignment probability

$$\begin{aligned}
\mathsf{P}(|T \cap S| < l) &\leq \mathsf{P}(\exists \mathbf{x}_i \text{ s.t. } |\overline{Y}(\mathbf{x}_i) - g(\mathbf{x}_i)| \geq \delta) \\
&= \mathsf{P}\left(\bigcup_{i=1,\ldots,k} [|\overline{Y}(\mathbf{x}_i) - g(\mathbf{x}_i)| \geq \delta] \right) \\
&\leq \sum_{i=1}^{k} \mathsf{P}(|\overline{Y}(\mathbf{x}_i) - g(\mathbf{x}_i)| \geq \delta) \\
&\leq k e^{-n\beta}.
\end{aligned}$$

The last inequality follows from (2.7). Therefore, the misalignment probability decays exponentially fast as simulation replication n increases, confirming the first basic idea that estimating order is easier than estimating the absolute objective value.

It is also worthwhile noticing that the analysis above does not impose the normality assumption as in the R&S algorithms in Sect. 2.3. A finite moment generating function is both sufficient and necessary to achieve the asymptotic exponential convergence rate with respect to order [20]. However, common distributions such as lognormal and certain Gamma distributions do not have finite moment generating functions. In such cases, proper truncations can be used to recover the exponential convergence rate [20].

To understand the benefit of goal softening, we assume that we randomly (and thus blindly) pick the set S from all k solutions. For simplicity, we only consider $l = 1$. Let $s = |S|$ and $t = |T|$. The misalignment probability is $\mathsf{P}(|T \cap S| = 0) = \binom{k-t}{s} / \binom{k}{s}$. So the AP is given by

$$\mathsf{P}(|T \cap S| \geq 1) = 1 - \mathsf{P}(|T \cap S| = 0)$$

$$= 1 - \binom{k-t}{s} \Big/ \binom{k}{s}$$
$$= 1 - \frac{(k-t)(k-t-1)\cdots(k-t-s+1)}{k(k-1)\cdots(k-s+1)}. \tag{2.8}$$

We can bound (2.8) by using the fact that

$$\frac{k-t-i}{k-i} = 1 - \frac{t}{k-i} \le 1 - \frac{t}{k},$$

for all $i = 0, 1, \ldots, s-1$. So we have

$$\mathsf{P}(|T \cap S| \ge 1) \ge 1 - \left(1 - \frac{t}{k}\right)^s. \tag{2.9}$$

Since $1 - t/k \le e^{-t/k}$, we can further bound (2.9) with the following inequality

$$\mathsf{P}(|T \cap S| \ge 1) \ge 1 - e^{-\frac{ts}{k}}. \tag{2.10}$$

The righthand side of (2.10) establishes the fact that as we soften our goal, i.e., increase t to make the "good enough" set T larger, or increase the size of the selected set s, the alignment probability converges exponentially to 1.

Instead of blindly picking the set S, one may run a few simulation replications on all $\mathbf{x} \in \Theta$ and choose S according to the sample mean $\overline{Y}(\mathbf{x})$. Under the assumption of a common additive Gaussian simulation error term $\mathcal{N}(0, \sigma^2)$ for all $\mathbf{x}$, when an equal number of simulation replications n is allocated to every $\mathbf{x} \in \Theta$, selecting the top s solutions ranked by $\overline{Y}(\mathbf{x})$ has the same lower bound on AP given in (2.10) as in the blind picking case under a Least Favorable Configuration (LFC) [42]. In an LFC, without loss of generality, we have $g(\mathbf{x}) = 0$ for all $\mathbf{x} \in S$ and $g(\mathbf{x}) = \Delta$ for all $\mathbf{x} \in \Theta \setminus S$. Notice that when $\Delta = 0$, it is equivalent to the blind picking case. Furthermore, if $\Delta > \overline{\Delta}$, where

$$\overline{\Delta} = \frac{\sigma}{\sqrt{n}} \left[\frac{1}{2} + \log\left(\frac{k-1}{\sqrt{2\pi}} \right) \right],$$

then a tighter lower bound on AP is [42]

$$\mathsf{P}(|T \cap S| \ge 1) \ge 1 - \frac{\max(t,s)}{|t-s|} e^{\left[-\min(t,s)(\Delta - \overline{\Delta}) \frac{\sqrt{n}}{\sigma} \right]}. \tag{2.11}$$

The new lower bound (2.11) shows that AP converges exponentially to 1 when

- the optimization goal is softened by increasing $\min(t,s)$;
- the difference in solution quality Δ between good and bad solutions is larger;
- simulation replications n increases.

The LFC represents the worst case scenario of AP and thus (2.10) is a universal lower bound on all problems. Given that the least favorable configuration is hardly the case in real problems, using $\overline{Y}(\mathbf{x})$ to choose the set S can achieve much higher alignment probability than the probability given in (2.8) when S is chosen randomly. However, unless there is structural information about the problem, such as an LFC configuration with Δ difference, there is no readily available tighter lower bound on AP other than (2.10). Nevertheless, for many practical engineering problems, one may conduct pilot experiments to gain knowledge about the problem and use the tables in [26] to identify the s that approximately achieves the required AP.

Therefore, to implement OO, one first determines a target AP and picks the set S. Once the set S is determined, one can then apply a R&S procedure to select the best solution in S. Because the set S is often much smaller than the feasible solution space Θ, R&S procedures are both effective and efficient. Furthermore, because the set S contains at least one good enough solution with a given AP and a R&S procedure can select the best solution from the set S with a given PCS, one can thus select the appropriate AP and PCS to ensure that the solution finally selected is a good enough solution of the original DOvS problem with a target probability.

2.5 Globally Convergent Random Search Algorithms

In this section we consider globally convergent algorithms designed for large, but still finite, feasible regions Θ. We focus on algorithms that exploit no structural information other than $|g(\mathbf{x})| < \infty$ and that we have a consistent estimator $\hat{g}(\mathbf{x})$ of $g(\mathbf{x})$ for all $\mathbf{x} \in \Theta$.

The methods described here can be broadly characterized as globally convergent adaptive random search (GCARS). We start by defining a high-level GCARS algorithm that contains the key features found in the research literature; we then discuss some of the possible choices for these features and their consequences. Finally, we present several specific algorithms that illustrate these choices.

Let $\Theta^\star \subset \Theta$ be the set of globally optimal solutions, which is guaranteed to be non-empty since $|\Theta| < \infty$. Further, let $m = 0, 1, 2, \ldots$ be the index of the number of iterations of the algorithm, and let $\mathbf{x}^\star_m$ be the solution that the algorithm would report as optimal if stopped at the end of iteration m. We are interested in algorithms that provide one of the following convergence guarantees:[1]

$$\lim_{m\to\infty} \mathsf{P}(\mathbf{x}^\star_m \in \Theta^\star) = 1 \text{ or}$$

$$\mathsf{P}\left(\lim_{m\to\infty} \mathbf{1}\{\mathbf{x}^\star_m \in \Theta^\star\} = 1\right) = 1,$$

i.e., convergence in probability or w.p.1, respectively.

[1] One can also define convergence of $\hat{g}(\mathbf{x}^*_m)$, the estimated optimal value, but we do not do so here. See for instance Andradóttir [1].

On iteration m of our generic algorithm, there is an estimation set $\mathscr{E}_m \subset \Theta$ containing the solutions that will be simulated to estimate, or refine the estimate of, $g(\mathbf{x})$ for all $\mathbf{x} \in \mathscr{E}_m$. From iteration to iteration the algorithm retains some memory of what is observed through estimation; let $\mathscr{M}_m \subset \Theta$ denote the set of solutions on which information is retained through iteration m. For the moment we are intentionally vague about what the information is, but it could be as little as the identity of $\mathbf{x}^\star_m$, or as much as a record of all solutions that have ever been estimated, the order in which they were encountered, and all of the observations collected on each of them.

Solutions in the estimation set are simulated; how much simulation effort is expended depends on a simulation allocation rule, denoted $\text{SAR}_m(\mathscr{E}_m|\mathscr{M}_m)$, which may depend on the estimation set, the solutions on which we retain information, and the iteration number. The result of estimation is that each solution $\mathbf{x} \in \mathscr{M}_m$ has a value, denoted $V(\mathbf{x})$, which may be an estimate of $g(\mathbf{x})$ or an indicator that $\mathbf{x}$ is, or is not, the current estimated optimal solution $\mathbf{x}^\star_m$.

To represent the "random" aspect of adaptive random search, let $F_m(\cdot|\mathscr{M}_m)$ be a probability distribution on $\mathbf{x} \in \Theta$ that may depend on the iteration m and information on the solutions in $\mathscr{M}_m$. Given these components, the generic GCARS algorithm is as follows:

Generic GCARS Algorithm

Initialization: Set $\mathscr{M}_0 = \emptyset$ and choose feasible solution $\mathbf{x}^\star_0$. Set the iteration index $m = 0$.

Sampling: Choose the estimation set $\mathscr{E}_m$ where some or all of the solutions are sampled from Θ according to $F_m(\cdot|\mathscr{M}_m)$.

Estimation: Apply the $\text{SAR}_m(\mathscr{E}_m|\mathscr{M}_m)$ to solutions $\mathbf{x} \in \mathscr{E}_m$ in the estimation set.

Iteration: Update $V(\mathbf{x})$ for all $\mathbf{x} \in \mathscr{E}_m$ and choose $\mathbf{x}^\star_{m+1}$ as the solution with the best $V(\mathbf{x})$ value. Update the set $\mathscr{M}_{m+1}$, let $m = m+1$ and go to Sampling.

Notice that the generic GCARS algorithm contains no stopping rule, which is appropriate for proving asymptotic convergence. In practice stopping may occur, for instance, when a computation budget is exhausted or when progress appears too slow. Unless all solutions in Θ are actually simulated, or we have structural information on g, it is not possible to stop a GCARS algorithm with any statistical guarantee that $\mathbf{x}^\star_m$ is an optimal solution.

We now describe several different ways that the steps of GCARS might be accomplished. Let $\mathscr{V}_m$ be the set of solutions that have been visited by the algorithm through iteration m. By "visited" we mean that the solutions in $\mathscr{V}_m$ have been simulated during one or more iterations; therefore, $\mathscr{V}_m = \cup_{j=0}^m \mathscr{E}_m$. A key requirement for GCARS that exploit no structural information about g is that

$$\mathscr{V}_m \xrightarrow{m\to\infty} \Theta \text{ w.p.1}, \tag{2.12}$$

i.e., all solutions will eventually be visited. Clearly this is a strong condition, and one that will not be realized in practice when Θ is very large. Therefore, the focus of

algorithm design is often on being as aggressive as possible in exploring promising areas of Θ while still maintaining Condition (2.12) as well as others.

The burden of insuring (2.12) falls primarily on F_m. Three types of distributions are common.

- A distribution $F_m(\cdot|\mathscr{M}_m)$ that puts positive probability on a small number of feasible solutions in a neighborhood of $\mathbf{x}^\star_m$. When this is the case, then F_m and the neighborhood structure on which it is defined must connect Θ so that any solution is reachable from any other solution after a sufficient number of iterations.
- A distribution $F_m(\cdot|\mathscr{M}_m)$ that puts positive probability on a "promising" subset $\Theta^m \subset \Theta$ that may be large or small, but is not necessarily a neighborhood of $\mathbf{x}^\star_m$. Typically these distributions attempt to use the memory $\mathscr{M}_m$ in an intelligent way to concentrate the search.
- A distribution $F_m(\cdot|\mathscr{M}_m)$ that puts positive probability on all of Θ, but may change as a function of m and $\mathscr{M}_m$. In this case the search is always global, although it may become probabilistically focused on promising regions.

For instance, the Stochastic Ruler algorithm [73] (described below) takes $\mathscr{M}_m = \{\mathbf{x}^\star_m\}$, and only the identity of the current estimated optimal solution is retained. The estimation set is $\mathscr{E}_m = \{\mathbf{x}^\star_m, \mathbf{x}'\}$, where $\mathbf{x}' \sim F_m(\cdot|\mathbf{x}^\star_m)$, and $F_m(\cdot|\mathbf{x}^\star_m)$ puts positive probability only on a neighborhood of $\mathbf{x}^\star_m$. When the support of F_m is a small local neighborhood of $\mathbf{x}^\star_m$, then solution sampling is typically easy. And because the estimation set is a single neighbor, $\mathbf{x}^\star_{m+1}$ is one of $\mathbf{x}^\star_m$ and $\mathbf{x}'$. Algorithms built in this way require very low memory, but need increasing effort per iteration to converge to an optimal solution. In other words, $\text{SAR}_m(\mathscr{E}_m|\mathbf{x}^\star_m)$ must prescribe longer and longer simulation runs as m increases.

The Nested Partitions algorithm [63, 64] (also described below) takes $\mathscr{M}_m = \mathscr{V}_m$, and also retains some measure of the value of each solution visited. Typical choices for the value are $V(\mathbf{x}) = C(\mathbf{x})$, a count of the number of times that $\mathbf{x}$ has been visited by the algorithm through iteration m; or $V(\mathbf{x}) = \overline{Y}(\mathbf{x}; n(\mathbf{x}))$, the cumulative sample mean of the $n(\mathbf{x})$ observations of solution $\mathbf{x}$ for all $\mathbf{x} \in \mathscr{V}_m$. In these algorithms,

$$\mathbf{x}^\star_{m+1} = \text{argmax}_{\mathbf{x}\in\mathscr{V}_m} C(\mathbf{x}) \text{ or} \tag{2.13}$$

$$\mathbf{x}^\star_{m+1} = \text{argmin}_{\mathbf{x}\in\mathscr{V}_m} \overline{Y}(\mathbf{x}; n(\mathbf{x})). \tag{2.14}$$

In the Nested Partitions algorithm, $F_m(\cdot|\mathscr{M}_m)$ assigns substantial probability to a subset $\Theta^m \subset \Theta$ that shows promise of containing good solutions based on previous iterations, but also some to $\Theta \backslash \Theta^m$ to insure convergence. Depending on how these subregions are formed, sampling $\mathbf{x}$ from Θ^m may be easy or computationally challenging. When the estimated optimal is defined by (2.14) or (2.13) it is not essential that SAR_m increase the simulation effort in m. For instance, when the cumulative average (2.14) is employed, and $\overline{Y}(\mathbf{x}; n(\mathbf{x}))$ satisfies a strong law of large numbers, then SAR_m need only assure that if $\mathbf{x}$ is in the estimation set infinitely often, then it will receive an infinite amount of simulation effort.

We now look at four GCARS algorithms in more detail to gain insight into the strengths and weaknesses of various approaches. To simplify the presentation and statement of results, we assume that there is a unique globally optimal solution $\mathbf{x}^\star$.

For the Stochastic Ruler algorithm, we need an unbiased estimator $\hat{g}(\mathbf{x})$ of $g(\mathbf{x})$, whose distribution need not change as a function of the iteration m; e.g., it could be $\overline{Y}(\mathbf{x};n)$ with n fixed. We also need constants a and b such that $\mathsf{P}(a \leq \hat{g}(\mathbf{x}) \leq b) = 1$ for all $\mathbf{x} \in \Theta$.

Stochastic Ruler Algorithm

Step 0. Choose a and b such that $\mathsf{P}(a \leq \hat{g}(\mathbf{x}) \leq b) = 1$ 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 t_m such that $t_m \to \infty$ as $m \to \infty$. Select an initial solution $\mathbf{x}_0^\star$ and set $m = 0$.

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

Step 2. For $i = 1$ to t_m 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
    $\mathbf{x}_{m+1}^\star = \mathbf{x}_m^\star$; go to Step 3
  endif
 Next i
 $\mathbf{x}_{m+1}^\star = \mathbf{x}'$

Step 3. $m = m+1$; go to Step 1

The Stochastic Ruler algorithm works because it insures that the search is attracted to $\mathbf{x}^\star$ from which it is difficult to leave. Specifically, the probability of rejecting the candidate solution $\mathbf{x}'$ at Step 2 is minimized at $\mathbf{x}^\star$; furthermore, the transition probability into $\mathbf{x}^\star$ is greater than out of $\mathbf{x}^\star$. And even though candidate solutions are generated from a stationary discrete-time Markov chain, the implied transition matrix that describes the movement from solution to solution is irreducible, aperiodic and finite, and its steady-state probabilities degenerate to a distribution putting probability 1 on $\mathbf{x}^\star$ as $m \to \infty$. Thus the convergence is in probability.

The algorithm is elegant and compact (since it retains no past data), but it is not adaptive and requires increasing effort from iteration to iteration in Step 2. As a result its performance in practice is often poor.

When memory of visited solutions is not a limitation, Andradóttir [1] showed 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 search. This approach makes almost sure convergence of the algorithms easy to prove via the strong law of large numbers, provides a better estimate of the true value of the selected solution whenever the algorithm terminates since information on it is accumulated, and tends to yield better empirical performance. Since 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. The next two algorithms we describe provide better performance than the Stochastic Ruler algorithm by being more adaptive and retaining the cumulative sample means.

The principles of branch and bound have had an important influence on DOvS, even though classical branch and bound techniques such as relaxing integrality constraints to bound the potential of a partition of the feasible region are not possible. We first describe the stochastic branch-and-bound method (SB&B) of Norkin et al. [51,52], and then show how it leads to the widely used Nested Partitions (NP) algorithms of Shi and Ólafsson [63, 64] and Pichitlamken and Nelson [58]. This development is based on Nelson [48].

To describe a simplified version of SB&B, let $\{\Theta^p\}$ be subsets of Θ creating a partition $\mathscr{P}$. Define the value of the optimal solution restricted to Θ^p by

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

Clearly $g(\mathbf{x}^\star) = \min_{\Theta^p\in\mathscr{P}} g^\star(\Theta^p)$. Suppose that there exist two bounding functions ℓ and u defined on subsets of Θ such that

- $\ell(\Theta^p) \le g^\star(\Theta^p) \le u(\Theta^p)$
- $u(\Theta^p) = g(\mathbf{x}')$ for some $\mathbf{x}' \in \Theta^p$
- If $|\Theta^p| = 1$ then $\ell(\Theta^p) = g^\star(\Theta^p) = u(\Theta^p)$.

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 w.p.1,

$$\lim_{m\to\infty} L_m(\Theta^p) = \ell(\Theta^p),$$
$$\lim_{m\to\infty} U_m(\Theta^p) = u(\Theta^p).$$

Under these assumptions, a SB&B algorithm is the following:

Stochastic Branch and Bound Algorithm

Step 1. Set $m = 0$, $\mathscr{P}_0 = \Theta$ and generate $L_m(\Theta)$ and $U_m(\Theta)$.
Step 2. Set

$$\Theta_m = \operatorname{argmin}\{L_m(\Theta^p) : \Theta^p \in \mathscr{P}_m\}$$
$$\mathbf{x}^\star_m \in \operatorname{argmin}\{U_m(\Theta^p) : \Theta^p \in \mathscr{P}_m\}.$$

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

As the algorithm progresses, better estimates are obtained of the bounding functions, and the partition with the best lower bound is partitioned finer and finer. Under mild conditions $\mathbf{x}_m^\star$ converges w.p.1 to $\mathbf{x}^\star$.

There are two practical barriers to the application of SB&B. First, there needs to be bounding functions ℓ and u and convergent estimators of them; see [24, 51, 52] for some specific stochastic optimization problems where this is the case. A second barrier is the computing overhead needed to retain and refine a larger and larger partition structure as the algorithm progresses, since no partition is ever eliminated from consideration as in deterministic branch and bound.

Notice, however, that for any subset Θ^p, it is trivially true that

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

$$L_m(\Theta^p) = \min_{\mathbf{x}\in\Theta^p} \overline{Y}(\mathbf{x}; n(\mathbf{x}))$$

satisfy the required conditions, provided $n(k)$, the cumulative number of replications of solution $\mathbf{x}$ through iteration k, increases. In other words, the smallest objective function value in a partition is a (tight) lower bound, and a consistent estimator of it is the smallest sample mean provided all solutions in the partition are simulated infinitely many times. But it is also true that the estimator

$$\hat{L}_m(\Theta^p) = \min_{\mathbf{x}\in\mathscr{X}^p(m)} \overline{Y}(\mathbf{x}; n(\mathbf{x}))$$

works provided $\mathscr{X}^p(m)$ is a randomly sampled subset of solutions from Θ^p that converges to Θ^p as $m \to \infty$. Therefore, $\hat{L}_m(\Theta^p)$ is a sampling-based lower bound that is available for any problem, which 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, $\mathscr{P}_{m+1}$, to be

$$\mathscr{P}_{m+1} = (\Theta \backslash \Theta_m) \cup \mathscr{P}'_m.$$

In words, we maintain only the most recently refined partition, and aggregate all other feasible solutions into a single "surrounding region." With these two refinements SB&B becomes a version of the NP method that is similar to Pichitlamken and Nelson [58].

NP uses a very straightforward adaptation: sample solutions more intensely in the partition that has most recently provided an apparently good solution, but continue to sample solutions from the surrounding region in case the global optimal is in it. The effectiveness of both SB&B and NP can be enhanced by making good decisions about which region to partition further, and how many solutions to sample from each partition. Shi and Ólaffson [64] describe embedding ranking and selection (Sect. 2.3) or ordinal optimization (Sect. 2.4) into NP to increase the likelihood that it partitions a region with good solutions. Xu and Nelson [71] suggest using a more sophisticated sampling-based bound than $\hat{L}_m$, one that is based on an

empirical Chebyshev inequality, to guide the solution sampling effort allocated to each partition in SB&B.

A typical strategy for GCARS is to exploit (search intensively) regions of Θ that appear to have good solutions, while still maintaining enough global exploration to be sure to capture $\mathbf{x}^\star$ in the limit. And since $g(\mathbf{x})$ can only be estimated, solutions must not only be visited, they must be estimated with less and less error in the limit to allow convergence. Prudius and Andradóttir [60] proposed a framework called balanced explorative and exploitative search with estimation (BEESE), which keeps global exploration, local exploitation and solution estimation in play by switching back and forth among them. Specifically, BEESE uses a `Global` probability distribution that places positive probability on all elements of Θ for exploration; a family of `Local` probability distributions that 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 probability 1 global convergence of an algorithm that falls into the BEESE framework can be proved provided the `Global` search distribution satisfies certain conditions. The simplest version of BEESE, known as R-BEESE, has the following high-level structure:

R-BEESE

Step 1. Sample a solution $\mathbf{x}' \sim \texttt{Global}(\Theta)$ and estimate $g(\mathbf{x}')$.

Step 2. With probability q, take additional replications of the current sample best solution $\mathbf{x}^\star_m$ to refine the estimate of $g(\mathbf{x}^\star_m)$.
Else w.p. 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 a solution $\mathbf{x}' \sim \texttt{Local}(\mathbf{x}^\star_m)$ and estimate or refine the estimate of $g(\mathbf{x}')$.

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

The `Global` and `Local` distributions on Θ, and the switching probabilities p and q, have an impact on performance. Since p and q are hard to choose (and should probably evolve), Prudius and Andradóttir [60] also describe A-BEESE which makes the switching decisions dynamic and adaptive to the progress of the search. The BEESE framework provides a structure and conditions that guarantee global convergence, but within which smart heuristics can be employed.

The Stochastic Ruler algorithm generates new candidate solutions directly from a neighborhood of the previous candidate; NP samples solutions intensely from a promising region defined by constraints; while R-BEESE switches between sampling solutions from a static global distribution on Θ and a local distribution that concentrates around the current sample best. Another approach is to always generate candidate solutions from a global probability distribution over Θ, but one that adapts based on the performance of previous candidates. Therefore, the search is always global, but concentrates on promising areas by changing the global distribution. The final GCARS algorithm we describe is the Model Reference Adaptive Search (MRAS) algorithm of Hu et al. [34, 35], and in particular its

stochastic simulation counterpart SMRAS [36]. MRAS and SMRAS are closely related to the cross-entropy method [62] and the estimation of distribution algorithm [46], and thus demonstrates the principles of those algorithms as well.

To make the algorithm easier to state, consider a problem where the goal is maximization, $g(\mathbf{x}) > 0$ and (for the moment) g can be evaluated exactly, i.e., we have a zero-variance estimator of $g(\mathbf{x})$, so the optimization problem is deterministic.[2] 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. Therefore r induces a distribution on the random variable $g(\mathbf{X})$, the value of the objective function at $\mathbf{X}$, making quantities such as $\mathsf{E}_r[g(\mathbf{X})]$ well defined.

Under appropriate conditions, there exists a recursive sequence of reference distributions $\{r_m; m = 0, 1, 2, \ldots)$ on Θ with the property that [35]

$$\lim_{m \to \infty} \mathsf{E}_{r_m}[g(\mathbf{X})] = g(\mathbf{x}^\star).$$

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

$$r_{m+1}(\mathbf{x}) = \frac{g(\mathbf{x}) r_m(\mathbf{x})}{\sum_{\mathbf{x}' \in \Theta} g(\mathbf{x}') r_m(\mathbf{x}')}.$$

If we could generate samples from r_m, then we could empirically estimate r_{m+1}, and continue to do this until the reference distribution essentially degenerates onto $\mathbf{x}^\star$. Unfortunately, r_m may have no special structure, making sampling from it computationally difficult. Therefore, MRAS samples solutions from Θ using a convenient parametric distribution $f(\cdot; \boldsymbol{\beta}_m)$, where at each iteration, $\boldsymbol{\beta}_m$ minimizes the Kullback–Leibler divergence between the parametric distribution and the reference distribution. SMRAS adapts MRAS to DOvS problems by substituting simulation estimators for $g(\mathbf{x})$, increasing the precision of these estimators as the algorithm closes in on $\mathbf{x}^\star$. A high-level description of SMRAS is given below.

SMRAS Algorithm

Step 1. Choose initial distribution $f(\cdot; \boldsymbol{\beta}_0)$ that assigns positive probability to all of Θ; a mixing coefficient $\lambda \in (0, 1)$; an initial number of solutions to sample t_0; an initial simulation sample size $n_0 > 1$; a simulation allocation rule n_m; and set $m = 0$.

[2] Clearly any minimization problem on $g(\mathbf{x})$ can be formulated as a maximization problem. If an estimator $\hat{g}(\mathbf{x})$ of $g(\mathbf{x})$ could be negative, then MRAS/SMRAS maximizes $s(g(\mathbf{x}))$ instead, where s is a non-negative, strictly increasing function.

Step 2. Generate t_m candidate solutions from $\bar{f}(\cdot;\boldsymbol{\beta}_m) = (1-\lambda)f(\cdot;\boldsymbol{\beta}_m) + \lambda f(\cdot;\boldsymbol{\beta}_0)$ to fill the estimation set $\mathscr{E}_m = \left\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(t_m)}\right\}$.

Step 3. Simulate n_m i.i.d. observations $Y_1(\mathbf{x}), Y_2(\mathbf{x}), \ldots, Y_{n_m}(\mathbf{x})$ for each solution $\mathbf{x} \in \mathscr{E}_m$ and compute its sample mean $\overline{Y}(\mathbf{x}; n_m)$.

Step 4. Calculate a threshold γ for elite solutions.

Step 5. Determine new distribution parameter $\boldsymbol{\beta}_{m+1}$ by solving

$$\boldsymbol{\beta}_{m+1} = \arg\max_{\beta} \left\{ \frac{1}{t_m} \sum_{\mathbf{x} \in \mathscr{E}_m} \frac{[\overline{Y}(\mathbf{x}; n_m)]^m}{f(\mathbf{x}; \boldsymbol{\beta}_m)} w\left(\overline{Y}(\mathbf{x}; n_m)\right) \ln f(\mathbf{x}; \boldsymbol{\beta}) \right\}$$

where

$$w(y) = \begin{cases} 0, & \text{if } y \le \gamma - \varepsilon \\ \dfrac{y - \gamma + \varepsilon}{\varepsilon}, & \text{if } \gamma - \varepsilon < y < \gamma \\ 1, & \text{if } y \ge \gamma. \end{cases}$$

Step 6. Set $m = m+1$, choose new solution sample size t_m and go to Step 2.

Step 2.5 minimizes the empirical Kullback–Leibler divergence between the parametric distribution and the reference distribution. There are conditions on the growth of n_m and t_m necessary for convergence; and while the algorithm need not maintain memory of previously visited solutions (since β_{k+1} completely specifies the sampling distribution for the next iteration) simulation effort can be saved by retaining $\overline{Y}(\mathbf{x}; n(\mathbf{x}))$ and $n(\mathbf{x})$ for each visited solution, so that if a solution is revisited then only $n_m - n(\mathbf{x})$ additional replications need to be obtained. The evolving threshold in Step 2.5 is also important for algorithm performance; see [36].

Notice that SMRAS employs the mixture distribution $\bar{f}(\cdot;\boldsymbol{\beta}_m) = (1-\lambda)f(\cdot;\boldsymbol{\beta}_m) + \lambda f(\cdot;\boldsymbol{\beta}_0)$, where $f(\cdot;\boldsymbol{\beta}_0)$ forces the algorithm to keep a global perspective. As a result, the distribution can never degenerate to the optimal solution. What can be shown is that for certain choices of parametric distributions

$$\lim_{m\to\infty} \mathsf{E}_{\beta_m}[\mathbf{X}] = \mathbf{x}^*.$$

2.6 Locally Convergent Random Search Algorithms

In this section, we consider locally convergent DOvS algorithms designed for a finite but potentially large feasible region Θ. As in Sect. 2.5, we focus on general-purpose algorithms that only assume $\mathrm{Var}[Y(\mathbf{x})] < \infty$ and that we have a consistent estimator $\hat{g}(\mathbf{x})$ of $g(\mathbf{x})$ for all $\mathbf{x} \in \Theta$.

Recall from Sect. 2.2 that we use $N(\mathbf{x}) \subset \Theta$ to denote the local neighborhood of a solution $\mathbf{x} \in \Theta$. We define $\mathbf{x}$ as a locally optimal solution if $g(\mathbf{x}) \leq g(\mathbf{y})$ for all $\mathbf{y} \in N(\mathbf{x})$. Since this definition of local optimality depends on the definition of $N(\mathbf{x})$, we may have different sets of locally optimal solutions when different neighborhood structures are used. For notational simplicity, we do not explicitly mention this dependence on the definition of $N(\mathbf{x})$ in this section.

Let $\mathscr{L}$ denote the set of locally optimal solutions for the DOvS problem. We again use $\mathbf{x}_m^*$ to denote the solution that the DOvS algorithm would report as optimal if terminated at the end of iteration m. We are interested in algorithms that provide local convergence in probability or local convergence w.p.1:

$$\lim_{m\to\infty} \mathsf{P}(\mathbf{x}_m^\star \in \mathscr{L}) = 1,$$

$$\mathsf{P}\left(\lim_{m\to\infty} \mathbf{1}\{\mathbf{x}_m^\star \in \mathscr{L}\} = 1\right) = 1,$$

Similar to GCARS, an LCARS algorithm has the following key components: an estimation set $\mathscr{E}_m \subset \Theta$ containing the solutions that will be simulated to estimate, or refine the estimate of, $g(\mathbf{x})$ for all $\mathbf{x} \in \mathscr{E}_m$; a memory set $\mathscr{M}_m \subset \Theta$ containing information on a set of solutions simulated up to iteration m; a simulation allocation rule, denoted $\mathrm{SAR}_m(\mathscr{E}_m|\mathscr{M}_m)$, determining how much simulation effort is expended on each solution $\mathbf{x} \in \mathscr{E}_m$, which may depend on the memory set $\mathscr{M}_m$ and the iteration number m; and a sampling probability distribution $F_m(\cdot|\mathscr{M}_m)$ to control the adaptive random search process.

Unlike $F_m(\cdot|\mathscr{M}_m)$ in GCARS algorithms, which may put positive probability on all of Θ, $F_m(\cdot|\mathscr{M}_m)$ in LCARS algorithms typically only puts positive probability on a "promising" subset $\Theta^m \subset \Theta$, which may be a small neighborhood of $\mathbf{x}_m^*$ or a larger area that is believed to contain good solutions according to information in $\mathscr{M}_m$. When global convergence is not required, focusing sampling on "promising" areas can speed up the progress of random search considerably.

Similar to GCRS, LCARS algorithms converge in an asymptotic sense as simulation effort goes to infinity. But unlike its globally convergent counterpart, an LCARS algorithm can be, and should be, combined with a statistical procedure to test the local optimality of $\mathbf{x}_m^*$. Formally, the statistical local optimality test of a solution $\mathbf{x}_m^*$ is

$$\mathrm{H}_0 : g(\mathbf{x}_m^*) \leq \min_{\mathbf{x}\in N(\mathbf{x}_m^*)} g(\mathbf{x}) \quad \text{vs} \quad \mathrm{H}_1 : g(\mathbf{x}_m^*) > \min_{\mathbf{x}\in N(\mathbf{x}_m^*)} g(\mathbf{x}).$$

A local optimality test procedure controls the type I and type II errors and provides an implementable stopping rule for LCRS algorithms to terminate the search when a locally optimal solution is found. Such a capability is a very desirable improvement over GCRS, for which it is not possible to stop with any statistical guarantee that $\mathbf{x}_m^*$ is an optimal solution. This test can be rewritten in the following form:

$$\mathrm{P}(\text{declare}\mathbf{x}_m^* \text{locallyoptimal}) \geq 1-\alpha_L \qquad \text{if} g(\mathbf{x}_m^*) \leq \min_{\mathbf{x}\in N(\mathbf{x}_m^*)} g(\mathbf{x}),$$

$$\mathrm{P}(\text{declare}\mathbf{x}_m^* \text{notlocallyoptimal}) \geq 1-\alpha_L \qquad \text{if} g(\mathbf{x}_m^*) \geq \min_{\mathbf{x}\in N(\mathbf{x}_m^*)} g(\mathbf{x}) + \delta_L,$$

where α_L is the type I error and δ_L is the indifference zone parameter. This test is thus a special case of comparison with a standard, which is $\mathbf{x}_m^*$. Therefore, efficient sequential procedures such as that of Kim [38] can be used to perform this test.

The generic LCARS algorithm is identical to the generic GCARS algorithm presented in Sect. 2.5. We now look at two specific LCARS algorithms in more detail to learn how different components of the generic LCARS algorithm can be designed to improve the practical performance of an LCARS algorithm while preserving its local convergence property. We will pay special attention to the design of the "promising" subset $\Theta^m \subset \Theta$ and the sampling probability distribution $F_m(\cdot|\mathscr{M}_m)$.

Convergent Optimization via Most-Promising-Area Stochastic Search (COMPASS) [28] proposes a unique structure of the most promising area Θ^m: it includes all solutions that are closer to $\mathbf{x}_m^*$ than any other simulated solution. Let $\mathscr{V}_m$ denote the set of all solutions simulated through iteration m, and $N_m(\mathbf{x})$ be the total number of i.i.d. simulation replications a solution $\mathbf{x}$ has received up to iteration m. COMPASS starts with an initial feasible solution $\mathbf{x}_0 \in \Theta$ provided by the user and randomly samples t_m solutions (duplicates allowed) from Θ_m at each iteration.

COMPASS

Step 1. Set $m=0, \mathscr{V}_0 = \{\mathbf{x}_0\}, \mathbf{x}_0^* = \mathbf{x}_0$. Simulate $n_0(\mathbf{x}_0)$ i.i.d. observations for $\mathbf{x}_0$, set $N_0(\mathbf{x}_0) = n_0(\mathbf{x}_0)$, and calculate its sample mean $\overline{Y}_0(\mathbf{x}_0)$. Let $\Theta_0 = \Theta$.

Step 2. Let $m = m+1$. Sample t_m candidate solutions $\mathbf{x}_m^{(1)}, \mathbf{x}_m^{(2)}, \ldots, \mathbf{x}_m^{(t_m)}$ uniformly and independently from Θ_{m-1}. Let $\mathscr{V}_m = \mathscr{V}_{m-1} \bigcup \{\mathbf{x}_m^{(1)}, \mathbf{x}_m^{(2)}, \ldots, \mathbf{x}_m^{(t_m)}\}$. Determine $n_m(\mathbf{x})$ according to the SAR for every $\mathbf{x} \in \mathscr{V}_m$ and simulate $n_m(\mathbf{x})$ i.i.d. replications for every $\mathbf{x} \in \mathscr{V}_m$. Update $N_m(\mathbf{x})$ and $\overline{Y}(\mathbf{x})$ for every $\mathbf{x} \in \mathscr{V}_m$.

Step 3. Let $\mathbf{x}_m^* = \arg\min_{\mathbf{x}\in\mathscr{V}_m} \overline{Y}(\mathbf{x})$. Let $\Theta_m = \{\mathbf{x} : \mathbf{x} \in \Theta, ||\mathbf{x}-\mathbf{x}_m^*|| \leq ||\mathbf{x}-\mathbf{y}|| \ \forall \mathbf{y} \in \mathscr{V}_m, \mathbf{y} \neq \mathbf{x}_m^*\}$. Go to Step 2.

Since COMPASS does not aim for global convergence, it can focus search effort in the area Θ_m, which is changed adaptively at each iteration based on information collected on all simulated solutions $\mathbf{x} \in \mathscr{V}_m$. As the algorithm iterates, Θ_m shrinks quickly and guides the search towards a potential locally optimal solution. When more simulations reveal that $\mathbf{x}_{m-1}^*$ is no longer the optimal solution, COMPASS allows the construction of a new Θ_m and moves the search towards new areas.

COMPASS maintains the cumulative sample mean of each simulated solution $\mathbf{x} \in \mathscr{V}_m$, and thus the memory set $\mathscr{M}_m = \mathscr{V}_m$. The sampling distribution $F_m(\cdot|\mathscr{M}_m)$ is uniform on Θ_m and puts zero density on $\Theta \setminus \Theta_m$. The estimation set $\mathscr{E}_m$ is set to be $\mathscr{V}_m$, and the SAR$(\mathscr{E}_m|\mathscr{M}_m)$ requires that $n_m(\mathbf{x}) \to \infty$ as $m \to \infty$. From a practical point of view, increasing $n_m(\mathbf{x})$ fast can slow down the progress significantly as

$\mathscr{E}_m$ includes every visited solution $\mathbf{x} \in \mathscr{V}_m$. The original COMPASS algorithm thus increases $n_m(\mathbf{x})$ logarithmically.

As COMPASS iterates, the most promising area Θ_m eventually will only contain $\mathbf{x}^*_m$. When this happens, we may perform a statistical local optimality test on $\mathbf{x}^*_m$ and its neighbors. Although COMPASS converges to a locally optimal solution w.p.1 as $m \to \infty$, the statistical local optimality test provides the capability to stop the optimization with a rigorous statistical guarantee of local optimality. This is another significant advantage of LCRS algorithms compared to GCRS algorithms that are typically stopped either when computation budget is exhausted or progress has been too slow, and thus no guarantee can be provided on the performance of $\mathbf{x}^*_m$ when the algorithm is stopped.

COMPASS sets the estimation set $\mathscr{E}_m$ to the entire set of visited solutions. As COMPASS iterates, the size of $\mathscr{E}_m$ keeps increasing and simulating all solutions in $\mathscr{E}_m$ becomes a big computational burden. It has been shown by Hong and Nelson [29] that for COMPASS to converge to a locally optimal solution w.p.1, it only requires the simulation effort for solutions that define the most promising area Θ_m to go to infinity as $m \to \infty$. The constraint defining Θ_m can be written in the following form:

$$(\mathbf{x}^*_m - \mathbf{x}_i)^\mathsf{T} \left(\mathbf{x} - \frac{\mathbf{x}^*_m + \mathbf{x}_i}{2} \right) \geq 0, \quad \mathbf{x}_i \in \mathscr{V}_m.$$

Some of these constraints are inactive in the sense that removing them will not change Θ_m. To determine if a solution $\mathbf{x}_i$ defines an active constraint, Xu et al. [69] propose to solve the following linear program (LP)

$$\begin{aligned} &\min_{\mathbf{x}} \ (\mathbf{x}^*_m - \mathbf{x}_i)^\mathsf{T} \left(\mathbf{x} - \frac{\mathbf{x}^*_m + \mathbf{x}_i}{2} \right) \\ &\text{s.t.} \ \ (\mathbf{x}^*_m - \mathbf{x}_j)^\mathsf{T} \left(\mathbf{x} - \frac{\mathbf{x}^*_m + \mathbf{x}_j}{2} \right) \geq 0 \qquad \forall \mathbf{x}_j \in \mathscr{V}_m \setminus \{\mathbf{x}^*_m\}, j \neq i. \end{aligned}$$

The solution $\mathbf{x}_i$ defines an active constraint if and only if the objective function value is negative. The LP needs to be solved for every $\mathbf{x}_i \in \mathscr{V}_m$ to find all active solutions. This step is referred to as constraint pruning [69]. When simulation is very time-consuming, there is still substantial saving in computational cost via constraint pruning. Numerical experiments conducted in [69] show that performing constraint pruning every 50 iterations seems to give good practical performance.

Hong and Nelson [29] introduce a generic LCRS algorithm framework with rather mild conditions on the sampling and estimation steps to ensure local convergence. Xu et al. [70] propose another set of conditions that facilitate implementation of fast LCRS algorithms. Their results show that to achieve local convergence, it is sufficient for an LCRS algorithm to satisfy the following conditions on

$F_m(\cdot|\mathscr{M}_m)$, $\mathscr{E}_m$, and the sample allocation rule $\text{SAR}_m(\mathscr{E}_m|\mathscr{M}_m)$ on every $\mathbf{x} \in \mathscr{E}_m$. In the following, let $\mathscr{S}_m$ be the set of solutions sampled from Θ_m at iteration m using the sampling distribution $F_m(\cdot|\mathscr{M}_m)$. Their conditions are:

1. The sampling distribution $F_m(\cdot|\mathscr{M}_m)$ guarantees that $\Pr\{\mathbf{x} \in \mathscr{S}_m\} \geq \varepsilon$ for all $\mathbf{x} \in \mathscr{N}(\mathbf{x}^*_{m-1})$ for some $\varepsilon > 0$ that is independent of m.
2. The estimation scheme satisfies the following requirements:
   (a) $\mathscr{E}_m$ is a subset of $\mathscr{V}_m$;
   (b) $\mathscr{E}_m$ contains $\mathbf{x}^*_{m-1}$ and $\mathscr{S}_m$;
   (c) $n_m(\mathbf{x})$ is allocated such that $\min_{\mathbf{x}\in\mathscr{E}_m} N_m(\mathbf{x}) \geq 1$ for all $m = 1,2,\ldots$ and $\min_{\mathbf{x}\in\mathscr{E}_m} N_m(\mathbf{x}) \to \infty$ w.p.1 as $m \to \infty$.

These flexible conditions allow the construction of alternative most promising areas Θ_m, which has critical influence on the practical performance of an LCRS algorithm. Xu et al. [70] propose a hyperbox-shaped Θ_m and call the algorithm the Adaptive Hyperbox Algorithm (AHA).

Let $x^{(k)}$ be the kth coordinate, $1 \leq k \leq d$, of a visited solution $\mathbf{x} \in \mathscr{V}_m$. Set $l_m^{(k)} = \max_{\mathbf{x}\in\mathscr{V}_m, \mathbf{x}\neq\mathbf{x}^*_m}\{x^{(k)} : x^{(k)} < x_m^{*(k)}\}$ if it exists; otherwise, let $l_m^{(k)} = -\infty$. Also, let $u_m^{(k)} = \min_{\mathbf{x}\in\mathscr{V}_m, \mathbf{x}\neq\mathbf{x}^*_m}\{x^{(k)} : x^{(k)} > x_m^{*(k)}\}$ if it exists; otherwise, let $u_m^{(k)} = \infty$. The hyperbox containing $\mathbf{x}^*_m$ is $\mathscr{H}_m = \{\mathbf{x} : l_m^{(k)} \leq x^{(k)} \leq u_m^{(k)}, 1 \leq k \leq d\}$.

In words, $u_m^{(k)}$ and $l_m^{(k)}$ give the boundaries, along the kth coordinate direction, of the largest hyperbox that encloses $\mathbf{x}^*_m$ but has all other visited solution $\mathbf{x} \in \mathscr{V}_m$ either on the boundary or outside. Note that $u_m^{(k)}$ and $l_m^{(k)}$ is $\pm\infty$ when there is no other solution to provide the hyperbox boundary along the kth coordinate direction. This may arise when $\mathbf{x}^*_m$ is on the boundary, or when AHA has not visited enough solutions yet. Let $\mathscr{L}_m = (l_m^{(1)}, \ldots, l_m^{(d)})$ and $\mathscr{U}_m = (u_m^{(1)}, \ldots, u_m^{(d)})$.

AHA constructs its most promising area Θ_m by finding $\mathscr{H}_m$ and setting $\Theta_m = \mathscr{H}_m \bigcap \Theta$. This construction of Θ_m allows AHA to shrink the volume of Θ_m exponentially fast and thus scales up to higher-dimensional DOvS problems. Another advantage is that it is much less computationally expensive to identify $\mathscr{H}_m$ than to identify the set of active solutions for the COMPASS algorithm. Again, we denote the starting solution as $\mathbf{x}_0$.

AHA

Step 1. Set $m = 0, \mathscr{V}_0 = \{\mathbf{x}_0\}, \mathscr{E}_0 = \{\mathbf{x}_0\}, \mathbf{x}^*_0 = \mathbf{x}_0$. Simulate $n_0(\mathbf{x}_0)$ i.i.d. observations for $\mathbf{x}_0$, set $N_0(\mathbf{x}_0) = n_0(\mathbf{x}_0)$, and calculate its sample mean $\overline{Y}_0(\mathbf{x}_0)$. Let $\mathscr{U}_0 = \mathscr{L}_0 = \mathscr{H}_0 = \emptyset$, and $\Theta_0 = \Theta$.

Step 2. Let $m = m+1$. Sample t_m candidate solutions $\mathbf{x}_m^{(1)}, \mathbf{x}_m^{(2)}, \ldots, \mathbf{x}_m^{(t_m)}$ uniformly and independently from Θ_{m-1}. Remove any duplicates from $\mathbf{x}_m^{(1)}, \mathbf{x}_m^{(2)}, \ldots, \mathbf{x}_m^{(t_m)}$, and let $\mathscr{S}_m$ be the remaining set. Let $\mathscr{E}_m = \mathscr{S}_m \bigcup \{\mathbf{x}^*_{m-1}\}$. Determine $n_m(\mathbf{x})$ according to the SAR for every $\mathbf{x} \in \mathscr{E}_m$ and simulate $n_m(\mathbf{x})$ i.i.d. replications for every $\mathbf{x} \in \mathscr{E}_m$. Update $N_m(\mathbf{x})$ and $\overline{Y}(\mathbf{x})$ for every $\mathbf{x} \in \mathscr{E}_m$.

Step 3. Let $\mathbf{x}_m^* = \arg\min_{\mathbf{x}\in\mathscr{E}_m} \overline{Y}(\mathbf{x})$. Identify $\mathscr{U}_m$ and $\mathscr{L}_m$ and thus $\mathscr{H}_m$. Let $\Theta_m = \mathscr{H}_m \bigcap \Theta$. Go to Step 2.

It is straightforward to verify that AHA satisfies the convergence conditions on the sampling distribution $F_m(\cdot|\mathscr{M}_m)$ and the estimation scheme. Therefore, AHA converges to the set of locally optimal solutions w.p.1.

Both COMPASS and AHA use a uniform sampling distribution with support on the most promising area Θ_m. This choice is reasonable when there is no structural information about the problem inside the most promising area Θ_m. As an alternative, Hong et al. [32] propose uniformly sampling along coordinate directions inside Θ_m and illustrate with a special case how coordinate sampling may help increase the chance of finding a locally optimal solution inside Θ_m. Yet another approach by Sun et al. [65] proposes to use a Gaussian process model as the sampling distribution to balance exploration and exploitation. It can be effectively combined with an LCRS algorithm like COMPASS or AHA to improve its practical performance.

Another category of locally convergent DOvS algorithms extend the problem to the continuous domain via linear interpolation. The advantage is by doing so, one can apply efficient gradient-based line search methods. The R-SPLINE algorithm of Wang, Pasupathy, and Schmeiser [66, 67] works within a retrospective framework [10,33,55–57]. At each iteration m, the retrospective framework converts a stochastic problem into a deterministic problem by averaging across k_m sample paths (generated using common random numbers). Given the deterministic sample-path problem, R-SPLINE uses piecewise linear interpolation to extend the problem into the continuous domain, and gradient estimates can then be computed to perform a line search. R-SPLINE also conducts a neighborhood enumeration search after every line search step. As $k_m \to \infty$ with at least a logarithmic pace, R-SPLINE converges to a locally optimal solution w.p.1.

Stochastic approximation [37, 61] uses stochastic estimates of the gradient directly to guide a line search. Lim [43] also extends the discrete problem $g(\mathbf{x})$ into a continuous problem $\tilde{g}(\mathbf{x})$ via piecewise linear interpolation. The basic idea is to find a $\tilde{g}(\mathbf{x})$ that has the following properties:

1. Both $g(\mathbf{x})$ and $\tilde{g}(\mathbf{x})$ have the same set of locally optimal solutions.
2. It is relatively easy to compute unbiased estimates of $\tilde{g}(\mathbf{x})$.
3. Stochastic approximation converges to a locally optimal solution of $\tilde{g}(\mathbf{x})$.

When these conditions are satisfied, stochastic approximation can be used to solve the original DOvS problem. The algorithms in [43] assume simulation noise has zero mean and finite variance. For a one-dimensional problem, the algorithm requires that $g(\mathbf{x})$ has a unique local minimizer. In the multidimensional case, the algorithms require that $g(\mathbf{x})$ is $L^\natural$-convex or multimodular [47]. Multimodular or $L^\natural$-convex functions arise naturally in many important problems in inventory systems and queueing networks.

2.7 Algorithm Enhancements

R&S procedures have also been used in conjunction with other DOvS algorithms to improve their efficiency or to make a correct decision at the end of the optimization process. Boesel et al. [6] proposed a "clean up" R&S selection procedure that selects the best solution among all solutions evaluated by a DOvS algorithm and provides a fixed-width confidence interval for the objective function value of the best solution. Many search-based OvS algorithms select the best solution from a neighborhood. Pichitlamken et al. [59] designed a sequential procedure for that purpose. Since a DOvS algorithm often runs for many iterations and the algorithm may stop at any iteration, it is desirable to have a R&S procedure that guarantees the solution of the current iteration is the best among all visited solution. Hong and Nelson [30] designed such a procedure. In Xu et al. [69], they also used the comparison-with-a-standard procedure of Kim [38] to test the local optimality of a solution when solving DOvS problems.

2.8 Using Commercial Solvers

This section is based on material in Hong and Nelson [31], Chap. 12 of Banks et al. [3], and Nelson [48].

Most commercial simulation modeling software also includes an OvS tool; however, to the best of our knowledge none of these tools are based on the DOvS algorithms presented in this chapter, with the exception of the ranking and selection procedures that are found in a number of simulation packages. In addition, a free version of COMPASS called "Industrial Strength COMPASS" can be obtained from `www.iscompass.net`; with some effort it could be used in conjunction with commercial simulation modeling software, although it is most suitable for use with a lower-level programming language such as `C++`.

Instead of provably convergent DOvS algorithms, robust metaheuristics are the most common foundation for integrated OvS tools. A "robust metaheuristic" is an OvS procedure that does not depend on strong problem structure to be effective, and is somewhat tolerant of some sampling variability. Examples include genetic algorithms and tabu search. These integrated tools can be applied to problems with continuous, integer and categorical decision variables. Robust metaheuristics have been observed to be effective on difficult *deterministic* optimization problems, but they usually provide no performance guarantees for deterministic problems, and certainly not for OvS problems. The following three simple ideas can make them more effective in practice and help avoid the three types of errors described in Sect. 2.1.4.

2.8.1 Preliminary Experiment to Control Sampling Variability

It will often be up to the simulation user to determine how many replications are needed at each feasible solution examined by the heuristic. We know that convergence requires that 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. Therefore at the beginning of the search very little error control may be needed for the solver to identify good solutions and search directions, but later in the search this might not be the case. Unfortunately, some solvers use the same number of replications at all solutions visited, and do not revisit solutions to add more replications.

However, some OvS software does have an "adaptive" setting, meaning it adjusts the number of replications based on the variance of the simulation estimates. If this feature is available, then use it. When the user 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 in the interior. 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, when the decision that will be based on the DOvS results really matters, then waiting hours or even days for the *best* decision may well be worth it.

2.8.2 Restarting the Optimization

Even with infinite effort, robust metaheuristics may provide no guarantee that they converge to the optimal solution. Therefore, the chances of finding a very good, or even the best, solution is increased if the solver is run multiple times. Each optimization run should use different random number seeds or streams, and should start from different initial solutions if possible. Be sure to select starting solutions on the extremes of the solution space, in the center of the space, or even randomly generated. If you suspect that certain solutions will be good, include them as starting solutions also. *These runs can be made in parallel on different computers.* The inconvenience of initializing several optimization runs is worth it if it leads to a much better solution, and if all runs lead to the same solution then you can have higher confidence that you have found the best.

2.8.3 Statistical Clean Up After Search

After the optimization run or runs have completed, it is critical to perform a second set of statistically designed experiments on the apparent best solutions identified by the heuristic; we call these "clean-up experiments."

In an OvS problem you can never be sure you have found the optimal solution; this is an error that cannot be avoided unless you exhaust Θ, although restarting helps. The two other types of errors are avoidable: failing to recognize the best solution that actually *was* visited, and poorly estimating the performance of the solution that was 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 because there is a natural bias toward solutions that, by chance, received favorable simulation estimates. Therefore, it is prudent to perform a rigorous statistical analysis, using a ranking-and-selection procedure such as those described in Sect. 2.3, to decide which are the best or near-best of the solutions visited during the search. Include at least the top 5 % of the solutions encountered during the search in this controlled experiment. The ranking-and-selection procedures built into the packages are ideal for this purpose.

The "clean up" concept was introduced in [6], which extended NSGS to be able to start with solutions having unequal numbers of observations, as one would expect at the end of a DOvS run.

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

Acknowledgements This work was supported in part by the National Science Foundation 1099 under Grant CMMI-1233376, and by the Hong Kong Research Grants Council under Project 613011, 613012 and N_HKUST626/10.

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. Simulation optimization: integrating research and practice. *INFORMS Journal on Computing*, 14:216–219, 2002.
3. J. Banks, J. S. Carson, B. L. Nelson, and D. M. Nicol. *Discrete-Event System Simulation*. Prentice Hall, Inc., Upper Saddle River, NJ, 5th edition, 2010.
4. R. E. Bechhofer. A single-sample multiple decision procedure for ranking means of normal populations with known variances. *Annals of Mathematical Statistics*, 25:16–39, 1954.
5. R. E. Bechhofer, T. J. Santner, and D. Goldsman. *Design and Analysis of Experiments for Statistical Selection, Screening and Multiple Comparisons*. John Wiley, New York, 1995.

6. 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.
7. J. Branke, S. E. Chick, and C. Schmidt. Selecting a selection procedure. *Management Science*, 53:1916–1932, 2007.
8. J. A. Buzacott and J. G. Shantikumar. *Stochastic Models of Manufacturing Systems*. Prentice-Hall, Englewood Cliffs, NJ, 1993.
9. C.-H. Chen, J. Lin, E. Yücesan, and S. E. Chick. Simulation budget allocation for further enhancing the efficiency of ordinal optimization. *Discrete Event Dynamic Systems: Theory and Applications*, 10:251–270, 2000.
10. H. Chen and B. W. Schmeiser. Stochastic rooting-finding via retrospective approximation. *IIE Transactions*, 33: 259–275, 2001.
11. S. E. Chick. Subjective probability and Bayesian methodology. In S. G. Henderson and B. L. Nelson, editors, *Handbooks in Operations Research and Management Science: Simulation*. Elsevier, New York, 2006.
12. S. E. Chick and K. Inoue. New two-stage and sequential procedures for selecting the best simulated system. *Operations Research*, 49:732–743, 2001.
13. L. Dai. Convergence properties of ordinal comparison in simulation of discrete event dynamic systems. *Journal of Optimization Theory and Applications*, 91:363–388, 1996.
14. L. Dai and C.-H. Chen. Rates of convergence of ordinal comparison for dependent discrete event dynamic systems. *Journal of Optimization Theory and Applications*, 94:29–54, 1997.
15. M. H. DeGroot. *Optimal Statistical Decisions*. Wiley, New York, 1970.
16. P. I. Frazier. Decision-theoretic foundations of simulation optimization. *Wiley Encyclopedia of Operations Research and Management Sciences*. Wiley, New York, 2010.
17. P. I. Frazier and W. B. Powell. The knowledge-gradient stopping rule for ranking and selection. In S. J. Mason, R. R. Hill, L. Möench, O. Rose, T. Jefferson, and J. W. Fowler, editors, *Proceedings of the 2008 Winter Simulation Conference*, pages 305–312. IEEE, Piscataway, NJ, 2008.
18. P. I. Frazier, W. B. Powell, and S. Daynik. A knowledge-gradient policy for sequential information collection. *SIAM Journal on Control and Optimization*, 47:2410–2439, 2008.
19. P. I. Frazier, W. B. Powell, and S. Daynik. The knowledge gradient policy for correlated normal beliefs. *INFORMS Journal on Computing*, 21:599–613, 2009.
20. M. C. Fu and X. Jin. On the convergence rate of ordinal comparison of random variables. *IEEE Transactions on Automatic Control*, 46: 1950–1954, 2001.
21. D. Goldsman and B. L. Nelson. Comparing systems via simulation. In J. Banks, editor, *Handbook of Simulation*. John Wiley, New York, 1998.
22. S. S. Gupta. *On a Decision Rule for a Problem in Ranking Means*. PhD thesis, University of North Carolina, Chapel Hill, NC, 1956.
23. S. S. Gupta. On some multiple decision (ranking and selection) rules. *Technometrics*, 7:225–245, 1965.
24. W. J. Gutjahr, A. Hellmayr, and G. Ch. Pflug. Optimal stochastic single-machine-tardiness scheduling by stochastic branch-and-bound. *European Journal of Operational Research*, 117:396–413, 1999.
25. Y.-C. Ho, R. Sreenivas and P. Vakili. Ordinal optimization of discrete event dynamic systems. *Journal of Discrete Event Dynamic Systems*, 2:61–88, 1992.
26. Y.-C. Ho, Q.-C. Zhao and Q.-S. Jia. *Ordinal Optimization: Soft Optimization for Hard Problems*. Springer, New York, 2007.
27. L. J. Hong and B. L. Nelson. The tradeoff between sampling and switching: New sequential procedures for indifference-zone selection. *IIE Transactions*, 37:623–634, 2005.
28. L. J. Hong and B. L. Nelson. Discrete optimization via simulation using COMPASS. *Operations Research*, 54:115–129, 2006.
29. 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.

30. L. J. Hong and B. L. Nelson. Selecting the best system when systems are revealed sequentially. *IIE Transactions*, 39:723–734, 2007.
31. L. J. Hong and B. L. Nelson. A brief introduction to optimization via simulation. In M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin and R. G. Ingalls, editors, *Proceedings of the 2009 Winter Simulation Conference*, pages 75–85. IEEE, Piscataway, NJ, 2009.
32. L. J. Hong, B. L. Nelson and J. Xu. Speeding up COMPASS for high-dimensional discrete optimization via simulation. *Operations Research Letters*, 38:550–555, 2010.
33. J. Jin. *Retrospective Optimization of Stochastic Systems*. PhD thesis, Purdue University, West Lafayette, IN, 1998.
34. J. Hu, M. C. Fu, and S. I. Marcus. Stochastic optimization using model reference adaptive search. In M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, editors, *Proceedings of the 2005 Winter Simulation Conference*, pages 811–818. IEEE, Piscataway, NJ, 2005.
35. J. Hu, M. C. Fu, and S. I. Marcus. A model reference adaptive search method for global optimization. *Operations Research*, 55:549–568, 2007.
36. J. Hu, M. C. Fu, and S. I. Marcus. A model reference adapative search method for stochastic global optimization. *Communications in Information and Systems*, 8:245–276, 2008.
37. J. Kiefer and J. Wolfowitz. Stochastic estimation of the maximum of a regression function. *Annals of Mathematical Statistics*, 23:462–466, 1952.
38. S.-H. Kim. Comparison with a standard via fully sequential procedure. *ACM Transactions on Modeling and Computer Simulation*, 15:1–20, 2005.
39. S.-H. Kim and B. L. Nelson. A fully sequential procedure for indifference-zone selection in simulation. *ACM Transactions on Modeling and Computer Simulation*, 11:251–273, 2001.
40. S.-H. Kim and B. L. Nelson. Selecting the best system. In S. G. Henderson and B. L. Nelson, editors, *Handbooks in Operations Research and Management Science: Simulation*. Elsevier, New York, 2006.
41. S.-H. Kim and B. L. Nelson. On the asymptotic validity of fully sequential selection procedures for steady-state simulation. *Operations Research*, 54:475–488, 2006.
42. L.-H. Lee, T.-W. E. Lau, and Y.-C. Ho. Explanation of goal softening in ordinal optimization. *IEEE Transactions on Automatic Control*, 44:94–99, 1999.
43. E. Lim. Stochastic approximation over multidimensional discrete sets with applications to inventory systems and admission control of queueing networks. *ACM Transactions on Modeling and Computer Simulation*, 22:19:1–19:23, 2012.
44. J. Luo, L. J. Hong, B. L. Nelson, and Y. Wu. Fully sequential procedures for large-scale ranking-and-selection problems in parallel computing environments. Working paper, Department of Industrial Engineering and Logistics Management, Hong Kong University of Science and Technology, 2013.
45. S. Mahajan and G. Van Ryzin. Stocking retail assortments under dynamic consumer substitution. *Operations Research*, 49:334–351, 2001.
46. H. Mühlenbein and G. Paaß. From recombination of genes to the estimation of distributions I. Binary parameters. In H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature – PPSN IV*, pages 178–197. Springer Verlag, Berlin, Germany, 1996.
47. K. Murota. Note on multimodularity and *L*-convexity. *Mathematics of Operations Research*, 30:658–661, 2005.
48. B. L. Nelson. Optimization via simulation over discrete decision variables. In J. J. Hasenbein, editor, *TutORials in Operations Research*, 7:193–207. INFORMS, Hanover, MD, 2010.
49. B. L. Nelson, J. Swann, D. Goldsman, and W.-M. Song. Simple procedures for selecting the best simulated system when the number of alternatives is large. *Operations Research*, 49:950–963, 2001.
50. J. Nocedal and S. Wright. *Numerical Optimization*. Springer, New York, 2nd edition, 2006.
51. V. I. Norkin, Y. M. Ermoliev, and A. Ruszczyński. On optimal allocation of indivisibles under uncertainty. *Operations Research*, 46:381–395, 1998.
52. V. I. Norkin, G. Ch. Pflug and A. Ruszczyński. A branch and bound method for stochastic global optimization. *Mathematical Programming*, 83:425–450, 1998.

53. T. Osogami. Finding probably best systems quickly via simulations. *ACM Transactions onModeling and Computer Simulation*, 19:12:1–12:18, 2009.
54. E. Paulson. A sequential procedure for selecting the population with the largest mean from k normal populations. *Annals of Mathematical Statistics*, 35:174–180, 1964.
55. R. Pasupathy. *Retrospective-Approximation Algorithms for the Multidimensional Stochastic Rooting-Finding Problem*. PhD thesis, Purdue University, West Lafayette, IN, 2005.
56. R. Pasupathy and B. W. Schmeiser. Retrospective-approximation algorithms for the multidimensional stochastic rooting-finding problem. *ACM Transactions on Modeling and Computer Simulation*, 19:2:1–2:36, 2009.
57. R. Pasupathy. On choosing parameters in retrospective-approximation algorithms for stochastic rooting-finding and simulation optimization. *Operations Research*, 58:889–901, 2010.
58. J. Pichitlamken and B. L. Nelson. A combined procedure for optimization via simulation. *ACM Transactions on Modeling and Computer Simulation*, 13:155–179, 2003.
59. 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.
60. A. A. Prudius and S. Andradóttir. Balanced explorative and exploitative search with estimation for simulation optimization. *INFORMS Journal on Computing*, 21:193–208, 2009.
61. H. Robbins and S. Monro. A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407, 1951.
62. 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.
63. L. Shi and S. Ólafsson. Nested partitions method for stochastic optimization. *Methodology and Computing in Applied Probability*, 2:271–291, 2000.
64. L. Shi and S. Ólafsson. *Nested Partitions Method, Theory and Applications*. Springer, New York, 2009.
65. L. Sun, L. J. Hong, and Z. Hu. Optimization via simulation using Gaussian process-based search. In S. Jain, R. R. Creasey, J. Himmelspach, K. P. White, and M. Fu, editors, *Proceedings of the 2011 Winter Simulation Conference*, pages 4139–4150. IEEE, Piscataway, NJ, 2011.
66. H. Wang. *Retrospective Optimization of Discrete Stochastic Systems Using Simplicial Linear Interpolation*. PhD thesis, Purdue University, West Lafayette, IN, 2009.
67. H. Wang, R. Pasupathy, and B. W. Schmeiser. Integer-ordered simulation optimization using R-SPLINE: Retrospective search with piecewise-linear interpolation and neighborhood enumeration. *ACM Transactions on Modeling and Computer Simulation*, 23:17:1–17:24, 2013.
68. X. Xie. Dynamics and convergence rate of ordinal comparison of stochastic discrete-event systems. *IEEE Transactions on Automatic Control*, 42: 586–590, 1998.
69. J. Xu, B. L. Nelson, and L. J. Hong. Industrial Strength COMPASS: A comprehensive algorithm and software for optimization via simulation. *ACM Transactions on Modeling and Computer Simulation*, 20:1–29, 2010.
70. J. Xu, B. L. Nelson, and L. J. Hong. An adaptive hyperbox algorithm for high-dimensional discrete optimization via simulation problems. *INFORMS Journal on Computing*, 25:133–146, 2013.
71. W. L. Xu and B. L. Nelson. Empirical stochastic branch-and-bound for optimization via simulation. *IIE Transactions*, 45:685–698, 2013.
72. R. R. Wilcox. A table for Rinott's selection procedure. *Journal of Quality Technology*, 16:97–100, 1984.
73. D. Yan and H. Mukai. Stochastic discrete optimization. *SIAM Journal of Control and Optimization*, 30:594–612, 1992.

http://www.springer.com/978-1-4939-1383-1

Handbook of Simulation Optimization

Fu, M.C. (Ed.)

2015, XVI, 387 p. 18 illus., 9 illus. in color., Hardcover
ISBN: 978-1-4939-1383-1