A Combined Procedure for Optimization via Simulation

JUTA PICHITLAMKEN
Kasetsart University
and
BARRY L. NELSON
Northwestern University

We propose an optimization-via-simulation algorithm for use when the performance measure is estimated via a stochastic, discrete-event simulation, and the decision variables may be subject to deterministic linear integer constraints. Our approach—which consists of a global guidance system, a selection-of-the-best procedure, and local improvement—is globally convergent under very mild conditions.

Categories and Subject Descriptors: I.6.6 [**Simulation and Modeling**]: Simulation Output Analysis

General Terms: Experimentation; Theory

Additional Key Words and Phrases: Random search, ranking and selection, stochastic optimization

1. INTRODUCTION

We use the term *optimization via simulation* to refer to the problem of maximizing or minimizing the expected (long-run average) performance of a discrete-event, stochastic system that is represented by a computer simulation model. Typical optimization-via-simulation algorithms that are found in the research literature try to move in relatively improving directions while utilizing some form of randomization to escape from local optimal solutions. Andradóttir's [1998] tutorial discusses a number of such methods, including the stochastic ruler algorithm [Yan and Mukai 1992], variants of simulated annealing altered to accommodate randomness (e.g., Gelfand and Mitter [1989], Gutjahr and Pflug [1996], and Alrefaei and Andradóttir [1999]), and Andradóttir's [1995,

Authors' addresses: J. Pichitlamken, Department of Industrial Engineering, Kasetsart University, Bangkok, Thailand; B. L. Nelson, Department of Industrial Engineering & Management Sciences, Northwestern University, Evanston, IL 60208-3119; email: nelsonb@northwestern.edu.

1996] random search algorithms. Most of these algorithms can be shown to converge globally as the sampling effort increases. Fu [2002]—which is an update of Fu [1994]—provides an up-to-date tutorial on optimization-via-simulation research and practice. Swisher et al. [2004], which is notable for its broad reference list, extends and updates Jacobson and Schruben's [1989] survey. Goldsman and Nelson [1998] summarizes statistical procedures applicable in the simulation context.

In contrast to globally convergent methods, heuristics that seem to work well in practice have actually been implemented in commercial software. To provide some statistical validity for these heuristics, Boesel [1999] and Boesel et al. [2003a] suggest applying a ranking-and-selection procedure at the end of the search to allow the combined algorithm to make a correct-selection guarantee with respect to all solutions visited during the search; while useful, this "clean-up" approach does not ensure global convergence.

In this article, we attempt to bridge the gap between current research and practice. We propose a framework and specific algorithms for a broad class of discrete-decision-variable, optimization-via-simulation problems. Within our framework, it is possible to incorporate a variety of smart heuristics to achieve good empirical performance, while still maintaining a global convergence guarantee.

Specifically, our goal is to solve problems of the following generic form:

$$\max_{\mathbf{x}\in\Theta} \mu(\mathbf{x}) \tag{1}$$

when Θ is defined by the following constraints:

$$\begin{aligned} \sum_{i=1}^{q} a_{ij}x_i &\le b_j, \qquad j = 1,2,\ldots,p \\ 0 \le l_i \le x_i &\le u_i < \infty, i = 1,2,\ldots,q \\ l_i, x_i, u_i &\in \mathcal{Z}^+ \cup \{0\}, i = 1,2,\ldots,q \end{aligned} \tag{2}$$

where $\mathcal{Z}^+$ denotes the positive integers. Thus, we assume that the feasible region Θ is convex and finite. To avoid triviality, we also assume that Θ is nonempty. The finiteness of Θ allows us (conceptually) to index the solutions $\mathbf{x}$ and the corresponding performance measures as follows: $\Theta = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_v\}$ where v denotes the number of feasible solutions in Θ, and $\mu_i \equiv \mu(\mathbf{x}_i)$. Without loss of generality, we let the set of global optimal solutions be $\Theta^* = \{\mathbf{x}_{v^*}, \mathbf{x}_{v^*+1}, \ldots, \mathbf{x}_v\}$ where $1 \le v^* \le v$.

We assume that the function $\mu(\mathbf{x})$ is unknown, but we can estimate μ_i via a simulation experiment. The observed performance of solution i on replication p of the simulation is denoted by Y_{ip}, so that $\mu_i = \mathrm{E}[Y_{ip}]$. Let $\sigma_i^2 = \mathrm{Var}\,[Y_{ip}]$. Both the performance measure μ_i and its variance σ_i^2 are assumed finite for all $i \in \{1, 2, \ldots, v\}$. Further, the observed performance measures Y_{ip}, $p = 1, 2, \ldots$, are independent and identically distributed (i.i.d.), and independent of Y_{js} for $i \ne j$. We assume nothing else about the simulation output. These assumptions imply that our results are appropriate for any simulation experiment—terminating or steady state—that employs replications, and could conceivably

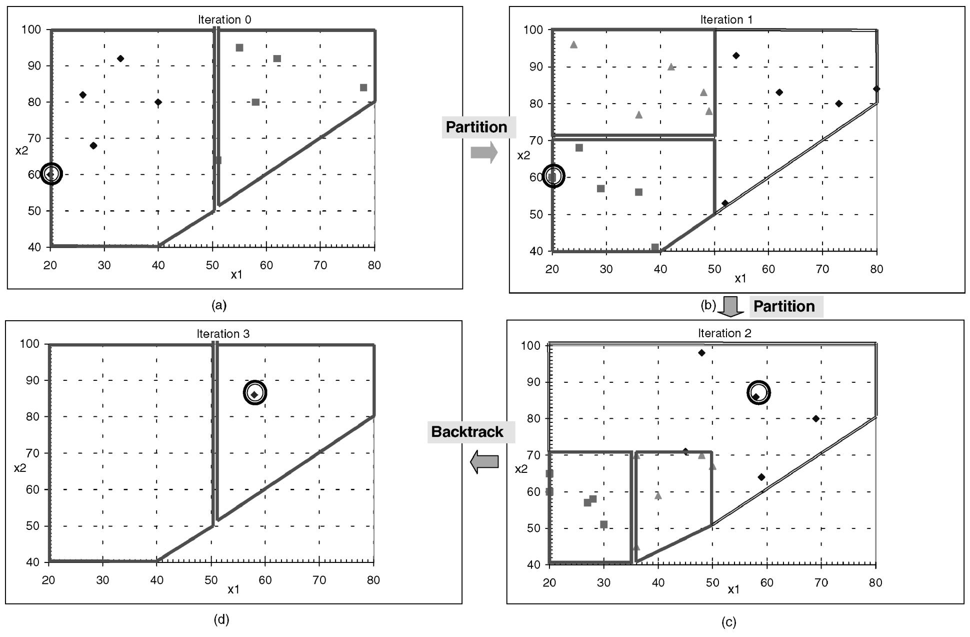

Fig. 1. Illustration of the NP approach for the (s, S) inventory problem.

be used within a single run of a steady-state simulation if appropriate initialization and batching strategies are employed.

Our optimization framework consists of a global guidance system, a selection-of-the-best procedure, and local improvement. The global guidance system ensures the convergence of the search so that, given sufficient time, it reaches and selects one of the optimal solutions. Specifically, we adopt the philosophy of Shi and Ólafsson's [2000] *Nested Partition* (NP) method. NP is based on identifying a sequence of "most-promising" subregions of Θ. When better solutions are found *inside* the current most-promising region, then the region is partitioned for finer exploration. On the other hand, when better solutions are found *outside* the current most-promising subregion, then NP backtracks to a superregion of it. The idea is to concentrate the computational effort where there appear to be good solutions but not be trapped locally.

Figure 1 illustrates the NP approach for an (s, S) inventory problem (Koenig and Law [1985]; see Sect. 4.1 for a brief description). Define $x_1 \equiv s$ and $x_2 \equiv S$. The constraints for this problem (in the form of (2)) are: $x_1 - x_2 \leq 0,\ 20 \leq x_1 \leq 80,\ 40 \leq x_2 \leq 100,\ x_1, x_2 \in \mathcal{Z}^+$.

Initially, the feasible region Θ is partitioned into some number of subregions, two in this illustration (see Figure 1(a)). Solutions are sampled within each subregion, their performance estimated via simulation, and the most-promising region is the one that contains the solution with the best (estimated) performance as indicated by the circled solution. The most-promising region is then partitioned further (see Figure 1(b)), and all other regions are aggregated into a "surrounding region." Again, solutions are sampled. If a subregion is most promising, then it is partitioned further (see Figure 1(c)) and the process continues. If the surrounding region has the best solution, then the search backtracks; in this illustration it backtracks to the original feasible region, Θ (see Figure 1(d)). When the most-promising region contains only a single solution, we refer to it as a *singleton region*.

The NP method provides the overall guidance system, but our implementation of it differs from Shi and Ólafsson's in two important ways: First, we select the most-promising region as the region that contains the best estimated solution using all data that have been obtained up to that point, while they use only samples obtained on the current NP iteration (this allows them to preserve a Markov property of the search). Second, at search termination they choose the solution that has been the most-promising singleton region most often; we choose the solution with the largest sample mean *accumulated over all visits to that solution*. Our choices permit a proof of almost sure convergence to an optimal solution under less restrictive conditions than Shi and Ólafsson. Of equal importance, our approach allows us to enhance empirical performance by incorporating *local improvement schemes* and statistical *ranking and selection procedures* without losing the convergence properties. We describe these enhancements next.

Because the NP method provides a *diversification* element (i.e., sampling the surrounding region), local improvement is intended to provide an *intensification* component. The idea is to improve performance on problems where $|\Theta|$ is large, but good solutions are clustered, or where $|\Theta|$ is large, but the response surface is smooth. Local improvement helps the search explore Θ more intensively near good solutions.

A *hill-climbing* (HC) algorithm constitutes our local-improvement scheme. We chose HC because it is intuitively simple: The current solution on hand is compared with some (or all) of its neighboring solutions, and the winner becomes the next solution. This neighborhood selection of the best is repeated until some stopping criterion is satisfied. Its simplicity aside, HC is also appealing because it is readily applicable in our problem setting where a neighborhood is easy to define. However, more sophisticated local-improvement schemes, including tabu search and response-surface methods, could easily be incorporated. Shi et al. [2001] incorporate local improvement into an NP-based method for solving a class of deterministic integer programming problems.

Each NP iteration, and each HC step, requires selecting the best solution from among a number of candidates (the sampled solutions for NP, and the neighboring solutions of the current best for HC). *Sequential Selection with Memory* (SSM) was designed to provide a highly efficient method for selecting the best—maximum or minimum expected performance—from among a small number of candidate solutions while controlling the chance of an incorrect selection. SSM is fully sequential with elimination, which means that it takes simulation outputs one at a time from the solutions under consideration and eliminates (ceases sampling) solutions as soon as they are shown to be inferior. SSM is specially designed for use in optimization algorithms that revisit solutions because it exploits whatever data have already been obtained. Under certain conditions, SSM guarantees to select the best, or a near-best, solution with a user-specified probability, where "near-best" means within a user-specified

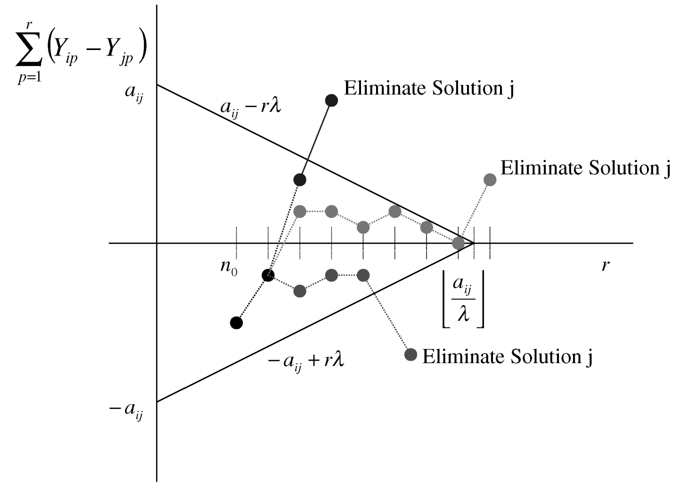

Fig. 2. Continuation region of SSM.

indifference level, $\delta > 0$. (See Shi and Chen [2000] for a similar idea based on the "Optimal Computing Budget Allocation" method.)

The *continuation region* for SSM (see Figure 2) illustrates the elimination step. Suppose we have only two solutions, $\mathbf{x}_i$ and $\mathbf{x}_j$. SSM continues taking simulation outputs as long as the sum of the difference between the simulation outputs of solutions i and j, $\sum_{p=1}^{r}(Y_{ip} - Y_{jp})$, stays within the triangular region. The sum can leave this region in three ways, as shown in the figure. The procedure is finite; at most, SSM takes one simulation output beyond the continuation region. Note that SSM is intended to aid optimization algorithms in making a correct selection over a subset of solutions, not to provide any correct-selection guarantee over all of Θ (see Pichitlamken and Nelson [2001] and Pichitlamken [2002] for details).

To implement the NP method for our chosen class of problems (1)–(2), we had to address two key technical issues. The first was to provide a partitioning scheme, which we solved using a strategy based on the Branch and Bound method. The second was to provide a method to randomly sample solutions from the (convex) subregions and the (nonconvex) surrounding region. For this purpose, we derived algorithms MIX-D and MIX-DS, respectively. Both algorithms generate a Markovian tour $\{\mathbf{X}_0, \mathbf{X}_1, \ldots, \mathbf{X}_T\}$, where $\mathbf{X}_t$ is a feasible solution in the region of interest.

Figure 3 illustrates Algorithm MIX-D on a two-dimensional Θ. From a starting solution inside the region, Algorithm MIX-D samples a direction that the Markovian tour will take (see Figure 3(a)). With the selected direction, MIX-D computes how far the tour can travel without leaving the region (see Figure 3(b)). In this illustration, the tour can proceed at most one step in the positive direction and three steps in the negative direction. MIX-D then uniformly samples the step length it will take; in Figure 3(c), MIX-D uniformly samples from $\{-1, 1, 2, 3\}$ and gets 1 as an outcome. MIX-D then traverses one step in the selected direction, and the current position of the tour is at $\mathbf{X}_1$. This process

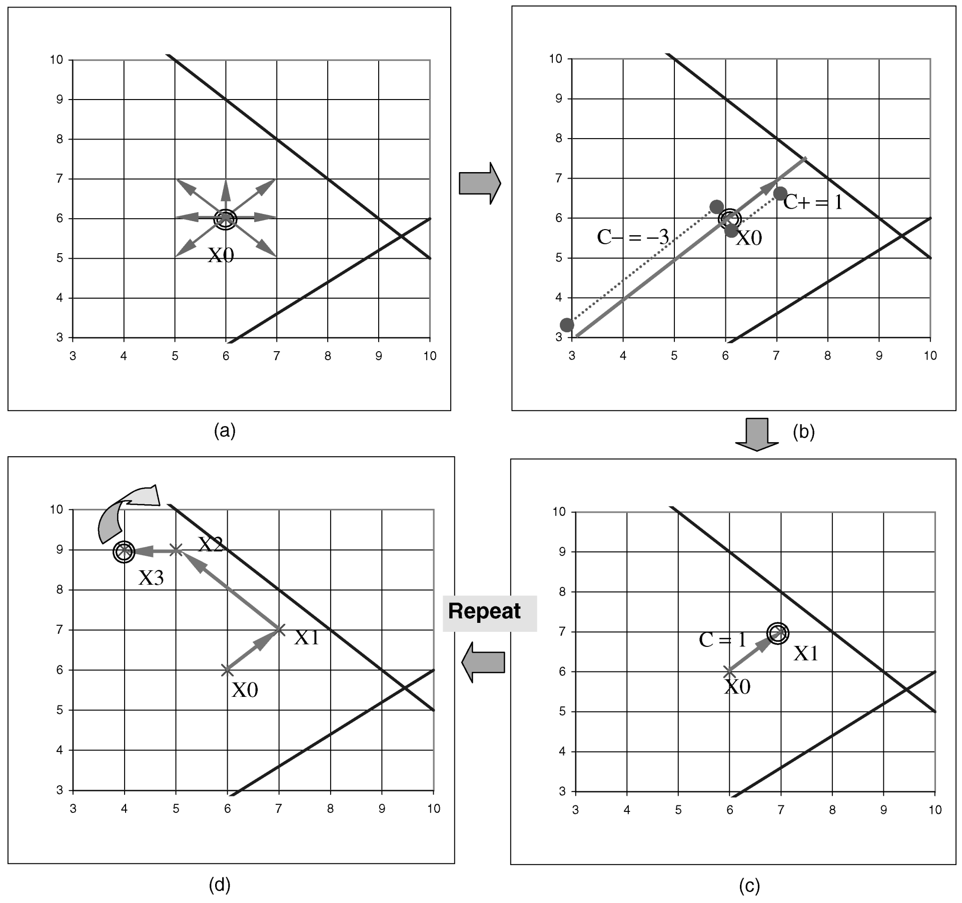

Fig. 3. Illustration of MIX-D.

is repeated for several steps, and the sampled solution returned by MIX-D is the current state of the Markov chain when it stops at step T (see Figure 3(d)). Algorithm MIX-DS is essentially Algorithm MIX-D, sampling over Θ, but with extra calculations to ensure that the generated Markov chain remains outside the most-promising region at all times.

A property of MIX-D and MIX-DS is that as the tour length T approaches infinity, the distribution of $\mathbf{X}_T$ is uniformly distributed over the subregion. Although the uniformity property of the solution sampling algorithms is not required to attain global convergence in our framework, it is desirable in practice: Since we assume no knowledge of the response surface $\mu(\mathbf{x})$, we count on the MIX-D and MIX-DS algorithms to ensure diversity, while the NP method focuses on promising solutions.

We call our combined scheme the *NP+SSM+HC Algorithm*. Our algorithm converges with probability 1 to a global optimal solution as the number of iterations approaches infinity. For the case of a unique optimum ($v^* = v$), the convergence rate is $O(k^{-1/2})$, where k is the number of iterations, and we have a Central Limit Theorem that allows us to form a confidence interval for the maximum performance measure μ_v.

The article is organized as follows: We give an overview of the NP+SSM+HC algorithm in Section 2, followed by detailed descriptions of each of its components in four subsections; any subsection can be skipped without loss of continuity. The global convergence properties are stated in Section 3. We compare our algorithm to other schemes via numerical examples in Section 4 and give some concluding comments in Section 5. All the proofs, and some algorithms, can be found in the Online Appendix. The complete empirical study is in Pichitlamken [2002].

2. NP+SSM+HC

We give a high-level description of NP+SSM+HC below.

(1) *Initialization.* Set the iteration counter $k = 1$, the current most-promising region $R_k = \Theta$, the number of observations on the ith solution $n_i(k) = 0$ for all $i \in \{1, 2, \ldots, v\}$, and the initial estimate of the optimal solution $\mathbf{x}_{\hat{i}^*_{k-1}}$ to a user-provided initial solution.

(2) *Search and Selection.* Repeat Steps (2)(a)–(2)(f) until the simulation effort (i.e., clock time or the number of simulation replications allowed) is exhausted:

  (a) *Partitioning.* If the current most-promising region R_k is not a singleton, then partition R_k into disjoint regions $R_{k1}, R_{k2}, \ldots, R_{k\omega(R_k)}$ (see Section 2.1). Let $M_k = \omega$ be the number of subregions. Then, if $R_k \neq \Theta$, aggregate the surrounding region; let $M_k = M_k + 1$ and $R_{kM_k} = \Theta \setminus R_k$.

  (b) *Sampling.* For each region $R_{k\ell}$, $\ell = 1, 2, \ldots, M_k$, randomly sample ϑ solutions from $R_{k\ell}$. (If $\mathbf{x}_{\hat{i}^*_{k-1}} \in R_{k\ell}$, include it as one of these ϑ sampled solutions from $R_{k\ell}$. See Sect. 2.2.) Aggregate *all* the sampled solutions $\mathbf{x}_i$ into a set through their indices i; let $\mathcal{S}_k$ denote the set of indices of sampled solutions.

  (c) *Selection of the Best Solution.* Take Δn_{free} observations of Y_{ip} from every solution $\mathbf{x}_i, i \in \mathcal{S}_k$. Use SSM or SSM(`REGION`) to select the best solution over $\mathcal{S}_k$, which we denote as $\hat{\mathbf{x}}^*(\mathcal{S}_k)$ (see Section 2.3). If the simulation effort is exhausted, go to the *Search termination* step.

  (d) *Algorithm Hill Climbing*. If the criterion for using HC is satisfied, perform Algorithm Hill Climbing with $\hat{\mathbf{x}}^*(\mathcal{S}_k)$ as a starting solution (see Section 2.4). Let $\mathbf{x}_{\hat{i}^*_k}$ be the solution deemed best by HC. If the simulation effort is exhausted, go to the *Search termination* step.

  (e) *Updating the Most-Promising Region.* If $\mathbf{x}_{\hat{i}^*_k} \in R_k$, then its subregion $R_{k\ell}$ that contains $\mathbf{x}_{\hat{i}^*_k}$ becomes the new most-promising region, R_{k+1}; otherwise, the search backtracks to the superregion of R_k, which can be either Θ or R_{k-1}. Increment $k = k + 1$.

  (f) *Restart.* Restart at iteration k if $R_{k-k_0+1} = R_{k-k_0+2} = \cdots = R_k$ (see Appendix A for the calculation of k_0) by letting $R_k = \Theta$; change the partitioning criterion (see (7) below).

(3) *Search Termination.* The best solution selected by NP+SSM+HC is the one with the maximum cumulative sample average; that is, the selected solution is $\mathbf{x}_{\hat{i}^*}$ where

$$\bar{Y}_i(r) \equiv \sum_{p=1}^{r} Y_{ip}/r \tag{3}$$

$$\hat{i}^* \equiv \arg\max_{1\le i\le v}\{\bar{Y}_i(n_i(k)) : n_i(k) > 0\}. \tag{4}$$

We further describe each component of NP+SSM+HC in the following sections (see Pichitlamken [2002] for complete details). A reader who is more interested in numerical results may skip these sections and continue on to Section 4.

To clarify the presentation that follows, let a *region*, which we will generically denote as τ, be a finite, convex subset of Θ characterized by:

$$\begin{aligned} \sum_{i=1}^{q} a_{ij}^{\tau} x_i &\le b_j^{\tau}, \qquad j = 1,2,\ldots,p^{\tau} \\ 0 \le l_i^{\tau} \le x_i &\le u_i^{\tau} < \infty,\ i = 1,2,\ldots,q \\ l_i^{\tau}, x_i, u_i^{\tau} &\in \mathcal{Z}^+ \cup \{0\}, i = 1,2,\ldots,q. \end{aligned} \tag{5}$$

A region is *infeasible* when there exists a constraint $j \in \{1,2,\ldots,p^{\tau}\}$ such that

$$\sum_{i:a_{ij}^{\tau}>0} a_{ij}^{\tau} l_i^{\tau} + \sum_{i:a_{ij}^{\tau}<0} a_{ij}^{\tau} u_i^{\tau} > b_j^{\tau}. \tag{6}$$

2.1 Partitioning Scheme

The goal is to partition a convex, feasible region τ of the form defined in (5) into disjoint subregions, each of which remains convex.

Input: A convex, feasible region τ of the form (5), number of subregions ω and branching criterion: `BIGGEST`, `SMALLEST` or `RANDOM` range (see (7)).
Output: $\omega' \le \omega$ disjoint feasible subregions $\tau_1, \tau_2, \ldots, \tau_{\omega'}$ of τ where

$$\bigcup_{\ell=1}^{\omega'} \tau_\ell = \tau \quad \text{and} \quad \tau_\ell \bigcap \tau_{\ell'} = \emptyset \text{ for } \ell \ne \ell'.$$

Procedure:

(1) Select a variable to branch on. Let i' be the index of the variable to be partitioned, where i' is defined as

$$i' \equiv \begin{cases} \text{argmax}_{1\le i\le q}\{u_i^{\tau} - l_i^{\tau}\} & \text{if criterion = BIGGEST RANGE} \\ \text{argmin}_{1\le i\le q}\{u_i^{\tau} - l_i^{\tau}\} & \text{if criterion = SMALLEST RANGE} \\ i \text{ with probability } 1/q \text{ for } i \in \{1,2,\ldots,q\} & \text{if criterion = RANDOM.} \end{cases} \tag{7}$$

(2) Keep the constraints and the bounds on other variables $x_i, i \ne i'$ in (5) unchanged. Divide the range of $x_{i'}$, $u_{i'}^{\tau} - l_{i'}^{\tau}$, into ω approximately equal intervals that can be characterized by the integer cutpoints $\{c_1 = l_{i'}^{\tau}, c_2, \ldots, c_{\omega'+1} = u_{i'}^{\tau}\}$, where $\omega' \le \omega$ is the number of subregions we can actually form. Table I shows how the cutpoints are computed. Let $[\chi]$ denotes rounding χ to the closest integer.

(3) Based on the cutpoints c_ℓ, $\ell \in \{1,2,\ldots,\omega'\}$, create a new region τ_ℓ. The constraints that define τ_ℓ consist of the constraints that define τ with one modification: the bound

Table I. Cutpoint computation for Algorithm Partition

```
$c_1 = l^{\tau}_{i'}$
$\epsilon = (u^{\tau}_{i'} - l^{\tau}_{i'})/\omega$
$\ell = 1$
while ($c_\ell \le u^{\tau}_{i'}$) do
    $\ell = \ell + 1$
    $c_\ell = \begin{cases} [c_{\ell-1} + \epsilon], & \text{if } [c_{\ell-1}+\epsilon] > c_\ell \\ [c_{\ell-1}+\epsilon]+1, & \text{otherwise} \end{cases}$
end while
```

constraint on $x_{i'}$ now becomes $c_\ell \le x_{i'} \le c_{\ell+1}$. That is, the constraint coefficients characterizing τ_ℓ are:

$$\begin{aligned} a^{\tau_\ell}_{ij} &= a^{\tau}_{ij} \text{ and } b^{\tau_\ell}_{j} = b^{\tau}_{j}, \forall i, j \\ l^{\tau_\ell}_{i} &= l^{\tau}_{i} \text{ and } u^{\tau_\ell}_{i} = u^{\tau}_{i}, \forall i \neq i' \\ l^{\tau_\ell}_{i'} &= c_\ell \text{ and } u^{\tau_\ell}_{i'} = c_{\ell+1}. \end{aligned} \tag{8}$$

After adding the tighter bound on $x_{i'}$, if region τ_ℓ is still feasible (in the sense of (6)), then update it by tightening variable bounds and eliminating redundant constraints. With the set of constraints (8), we recalculate the bounds for each variable $i, i = 1, 2, \ldots, q$ [Wolsey 1998];

$$l^{\tau_\ell}_{i} \leftarrow \max\left\{ l^{\tau_\ell}_{i}, \max_{j\in\{1,2,\ldots,p^{\tau}\}} \left\{ \frac{b^{\tau_\ell}_{j} - \sum_{h\neq i} a^{\tau_\ell}_{hj} x'_h}{a^{\tau_\ell}_{ij}} : a^{\tau_\ell}_{ij} < 0 \right\} \right\}$$

$$u^{\tau_\ell}_{i} \leftarrow \min\left\{ u^{\tau_\ell}_{i}, \min_{j\in\{1,2,\ldots,p^{\tau}\}} \left\{ \frac{b^{\tau_\ell}_{j} - \sum_{h\neq i} a^{\tau_\ell}_{hj} x'_h}{a^{\tau_\ell}_{ij}} : a^{\tau_\ell}_{ij} > 0 \right\} \right\}$$

where x'_h is set to $l^{\tau_\ell}_{h}$ if $a^{\tau_\ell}_{hj} > 0$ and $u^{\tau_\ell}_{h}$ if $a^{\tau_\ell}_{hj} < 0$. Finally, we remove a nonbinding constraint $j \in \{1, 2, \ldots, p^{\tau}\}$ if

$$\sum_{i:a^{\tau_\ell}_{ij}>0} a^{\tau_\ell}_{ij} u^{\tau_\ell}_{i} + \sum_{i:a^{\tau_\ell}_{ij}<0} a^{\tau_\ell}_{ij} l^{\tau_\ell}_{i} \le b^{\tau_\ell}_{j}.$$

The computational complexity of the partitioning scheme is linear with respect to the number of subregions ω, the number of variables q and the number of constraints p^{τ}.

On iteration k, after the most-promising region R_k is partitioned, and the surrounding region is aggregated, solutions are sampled from each subregion. We describe the solution sampling scheme next.

2.2 Solution Sampling Scheme

NP+SSM+HC requires sampling within the subsets of the most-promising region R_k, which are convex, and within the surrounding region, which is not convex. Our discrete-variable sampling algorithms extend Smith's [1984] Mixing Algorithm, which is for continuous spaces: Algorithm MIX-D samples an integer solution from a convex region of the form (5) (we will denote it as τ for simplicity), and Algorithm MIX-DS samples an integer solution from $\Theta \setminus R_k$. Algorithm MIX-DS is essentially Algorithm MIX-D, sampling over Θ, but with extra calculations to ensure that the generated Markov chain remains outside

R_k at all times. Therefore, we only provide the details of MIX-D here; MIX-DS can be found in Appendix B.

To characterize MIX-D, we need the following definitions: A *random tour* $\{\mathbf{X}_t, t \geq 0\}$ is a sequence of $\mathbf{X}_t \in \mathcal{Z}^q$. A *direction* is characterized by two end points, one of which is the origin. A *discrete interval* $[a, b]$, for $a \leq b$ and $a, b \in \mathcal{Z}$, is defined as

$$[a,b] \equiv \begin{cases} \{a\} & \text{for } a = b \\ \{a, a+1, \ldots, b\} & \text{for } a < b. \end{cases}$$

Algorithm MIX-D

Input: A closed convex region τ of the form (5), number of solutions to be sampled ϑ, length of warm-up period T, and a starting solution $\mathbf{x}_0 \in \tau$.
Output: ϑ random integer solutions which are feasible in τ.
Procedure:

(1) *Initialization*: Set $\mathbf{X}_0 = \mathbf{x}_0$, $t = 1$ and the set of sampled solutions $\mathcal{S} = \emptyset$.

(2) *Generating random solutions*:

(a) Sample a random *scaled* direction:

(i) Sample a random direction $\mathbf{U}_t \equiv (U_{1t}, U_{2t}, \ldots, U_{qt})$ uniformly over the set $\mathcal{D}(\mathbf{x}, \tau)$, where

$$\mathcal{D}(\mathbf{x}, \tau) \equiv \left\{\mathbf{U} : U_i \in \left[l_i^\tau - x_i, u_i^\tau - x_i\right], i = 1, 2, \ldots, q\right\} \setminus \{\mathbf{0}\}. \tag{9}$$

That is, for each dimension i, U_{it} is sampled uniformly from an integer interval $[l_i^\tau - x_i, u_i^\tau - x_i]$ independently from every other dimension but rejecting the zero vector $\mathbf{0}$.

(ii) Scale the direction $\mathbf{U}_t$ with the gcd of all $U_{it}, i = 1, 2, \ldots, q$, to yield the scaled direction $\mathbf{D}_t \equiv (D_{1t}, D_{2t}, \ldots, D_{qt})$ where

$$D_{it} \equiv \frac{U_{it}}{\text{gcd}_{1\leq r\leq q}\{U_{rt}\}}, \quad i = 1, 2, \ldots, q,$$

and $\text{gcd}_{1\leq i\leq q}\{a_1, a_2, \ldots, a_q\}$ is the greatest common divisor for all $a_i, i = 1, 2, \ldots, q$.

(b) Determine the set of possible integral increments:

$$\begin{aligned} \mathcal{C}(\mathbf{X}_{t-1}, \mathbf{D}_t, \tau) &\equiv \{C \in \mathcal{Z} : \mathbf{X}_{t-1} + C\mathbf{D}_t \in \tau \text{ and } C \neq 0\} \\ &= \left[C_t^-(\tau), C_t^+(\tau)\right] \setminus \{0\}, \end{aligned} \tag{10}$$

where

$$C_{jt}(\tau) \equiv \frac{\tilde{b}_j^\tau - \tilde{\mathbf{a}}_j^\tau \mathbf{X}_{t-1}}{\tilde{\mathbf{a}}_j^\tau \mathbf{D}_t} \tag{11}$$

$$C_t^-(\tau) \equiv \max_{1\leq j\leq p^\tau+2q} \{\lceil C_{jt}(\tau) : C_{jt}(\tau) < 0\rceil\} \tag{12}$$

$$C_t^+(\tau) \equiv \min_{1\leq j\leq p^\tau+2q} \{\lfloor C_{jt}(\tau) : C_{jt}(\tau) > 0\rfloor\}. \tag{13}$$

Note that $\lceil x\rceil$ is x rounded up to the next largest integer, and $\lfloor x\rfloor$ is x truncated to the next smallest integer. If $\mathcal{C}(\mathbf{X}_{t-1}, \mathbf{D}_t, \tau) = \emptyset$, then set $\mathcal{C}(\mathbf{X}_{t-1}, \mathbf{D}_t, \tau) = \{0\}$.

(c) Sample C_t uniformly on $\mathcal{C}(\mathbf{X}_{t-1}, \mathbf{D}_t, \tau)$.

(d) Update $\mathbf{X}_t$ and t:

$$\begin{aligned} \mathbf{X}_t &= \mathbf{X}_{t-1} + C_t\mathbf{D}_t, \\ t &= t + 1. \end{aligned} \tag{14}$$

(e) If $t < T$, go to Step (2)(a); otherwise, insert $\mathbf{X}_T$ into the set $\mathcal{S}$ if $\mathbf{X}_T \notin \mathcal{S}$. If $\mathcal{S}$ has ϑ solutions, return the set $\mathcal{S}$; otherwise, set $t = 0, \mathbf{X}_0 = \mathbf{X}_T$, and go to Step (2)(a).

It is worth noting that finding a solution $\mathbf{x}_0$ to initialize MIX-D can be an NP-Hard problem.

By construction, MIX-D generates solutions that constitute a Markov chain, a Markov Chain that follows a uniform distribution over τ as the length of the random tour T goes to infinity:

THEOREM 2.1. *The limiting distribution of the MIX-D Markov chain* $\{\mathbf{X}_t : t \geq 0\}$ *is uniform over* τ.

See the Online Appendix to this article for a proof.

After we sample solutions from all subregions, we use SSM to select the best. To make this article self-contained, we briefly describe this procedure in the next section (see Pichitlamken and Nelson [2001] and Pichitlamken [2002] for complete details).

2.3 Sequential Selection with Memory

In NP+SSM+HC, we use SSM for the selection of a new most-promising region for NP, and in determining when the HC algorithm has found an improved solution.

Without loss of generality, let the finite number of solutions under consideration be denoted by $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_\kappa\}$. SSM assumes the observations taken from $\mathbf{x}_i$, Y_{ip}, to be i.i.d. normally distributed with finite mean μ_i and variance σ_i^2. Although the normality assumption is used to establish the correct-selection guarantee of SSM, the probability 1 convergence of NP+SSM+HC to an optimal solution does not depend on the normality of the output data (see Section 3 below). For convenience of exposition, assume that the true means of the solutions are indexed such that $\mu_1 \leq \mu_2 \leq \cdots \leq \mu_\kappa$. The best solution is defined as the one with the largest mean, which is therefore μ_κ.

Our procedure guarantees to select $\mathbf{x}_\kappa$ with probability at least $1-\alpha$ whenever the difference between the best and the next-best solution is worth detecting:

$$\Pr\{\text{select } \mathbf{x}_\kappa\} \geq 1 - \alpha \text{ whenever } \mu_\kappa - \mu_{\kappa-1} \geq \delta. \tag{15}$$

The indifference-zone parameter is denoted by $\delta > 0$. Even when the indifference-zone condition is not satisfied ($\mu_\kappa - \mu_{\kappa-1} < \delta$), SSM still selects a "good" solution (one whose mean is within δ of μ_κ) with probability at least $1 - \alpha$ (see Pichitlamken [2002] for the proof).

In SSM, we sequentially take one observation at a time from *surviving* solutions, immediately followed by screening. To make this precise, we define some notation: Let $i \in \{1, 2, \ldots, \kappa\}$.

$$\begin{aligned} V &= \text{set of solutions we have "visited" before; } n_i \geq n_0 \text{ for } i \in V \\ V^c &= \text{set of solutions we see for the first time; } n_i < n_0 \text{ for } i \in V^c \\ &= \{1, 2, \ldots, \kappa\} \setminus V \\ N_{ij} &= \left\lfloor \frac{a_{ij}}{\lambda} \right\rfloor \end{aligned}$$

$$
\begin{aligned}
N_i &= \max_{j \neq i}\{N_{ij}\} \\
N &= \max_{1 \le i \le \kappa} N_i && (16) \\
\sigma_{ij}^2 &= \mathrm{Var}\,(Y_{ip} - Y_{jp}) \\
n_0 &= \text{minimum initial number of observations from any solution} \\
S_{ij}^2 &= \text{estimator of } \sigma_{ij}^2 \\
&= \frac{1}{n_0 - 1}\sum_{p=1}^{n_0}(Y_{ip} - Y_{jp} - [\bar{Y}_i(n_0) - \bar{Y}_j(n_0)])^2 && (17) \\
f &= n_0 - 1.
\end{aligned}
$$

Note that $N_i + 1$ is the maximum number of observations taken from $\mathbf{x}_i$.

Procedure SSM

(1) *Initialization.* For each $\mathbf{x}_i, i \in V^c$, take $n_0 - n_i$ observations ($n_0 \ge 2$), and set $n_i = n_0$. Compute $S_{ij}^2, \forall i \neq j$.

(2) *Procedure Parameters.* We choose λ and a_{ij} as follows:

$$\lambda = \frac{\delta}{2} \quad \text{and} \quad a_{ij} = \frac{f S_{ij}^2}{4(\delta - \lambda)}\left[\left(\frac{\kappa - 1}{2\alpha}\right)^{2/f} - 1\right]. \tag{18}$$

If $n_0 > N$ (as defined in (16)), stop and select the solution with the largest $\bar{Y}_i(n_i)$ (as defined in (3)) as the best. Otherwise, let $I = \{1, 2, \ldots, \kappa\}$ be the set of surviving solutions, set $r = n_0$, and proceed to *Screening*. From here on V represents the set of solutions on which we have obtained more than r observations, while V^c is the set of solutions with exactly r observations.

(3) *Screening.* Set $I^{\mathrm{old}} = I$ where

$$I = \left\{ i : i \in I^{\mathrm{old}} \text{ and } \mathcal{Y}_i \ge \max_{j \in I^{\mathrm{old}}, j \neq i} (\mathcal{Y}_j - a_{ij}) + r\lambda \right\}$$

$$\mathcal{Y}_j = \begin{cases} \sum_{p=1}^{r} Y_{jp} & \text{for } j \in V^c \\ r\bar{Y}_j(n_j) & \text{for } j \in V. \end{cases}$$

In essence, for $\mathbf{x}_i$ with $n_i > r$, we substitute $r\bar{Y}_i(n_i)$ for $\sum_{p=1}^{r} Y_{ip}$.

(4) *Stopping Rule.* If $|I| = 1$, then stop and report the only survivor as the best; otherwise, for each $i \in (I \cap V^c)$, take one additional observation from $\mathbf{x}_i$ and set $r = r + 1$. If $r = N + 1$, terminate the procedure and select the solution in I with the largest sample average as the best; otherwise, for each $i \in (I \cap V)$ with $n_i = r$, update V and V^c: $V^c = V^c \cup \{i\}$ and $V = V \setminus \{i\}$. Go to *Screening*.

SSM is a generic procedure that could be integrated into many optimization-via-simulation algorithms. SSM(REGION) is a refinement that is useful within an NP framework. SSM(REGION) intends to save simulation effort by terminating SSM when all surviving solutions belong to the *same* subregion. This is useful in an NP step where all we need to do is to identify the subregion that contains the best sampled solution, not necessarily the best solution itself.

Let $\mathcal{S}_k^*$ be the set of solution indices of the surviving solutions when SSM(REGION) terminates on iteration k. The best solution selected by

SSM(REGION) is:

$$\hat{\mathbf{x}}^*(\mathcal{S}_k) = \left\{\mathbf{x}_i : i = \mathrm{argmax}_{i \in \mathcal{S}_k^*} \bar{Y}_i(n_i)\right\}. \qquad (19)$$

After SSM or SSM(REGION) selects the best solution from the set of sampled solution, we attempt to improve it further with hill climbing, as described below.

2.4 Local Improvement

Algorithm Hill Climbing (HC) is essentially a greedy heuristic that iteratively moves from the current solution to one of its neighboring solutions until some stopping criterion is satisfied. In our algorithm, the available options for employing HC are: (a) do not perform HC at all, (b) perform HC on *all* iterations, or (c) perform HC if the *improvement* in successive iterations is "big enough," that is,

$$\left|\bar{Y}_{\hat{i}_k^*}\left(n_{\hat{i}_k^*}\right) - \bar{Y}_{\hat{i}_{k-1}^*}\left(n_{\hat{i}_{k-1}^*}\right)\right| > 2\delta, \qquad (20)$$

where δ is the indifference-zone parameter of SSM (see (15)).

Procedure Hill Climbing

(1) *Initialization.* Set $t = 0$ and $\mathbf{X}_0$ to a starting solution.

(2) *Search.* Repeat Steps (2)(a)–(2)(d) until the stopping criterion is satisfied (see Remark below):

   (a) *Neighborhood Construction.* For $\mathbf{X}_t = (X_{1t}, X_{2t}, \ldots, X_{qt})$, construct a hypercube: $\mathcal{B}(\mathbf{X}_t) \equiv \Pi_{i=1}^{q}[X_{it} - \xi, X_{it} + \xi]$, where $\xi \in \mathcal{Z}^+$. Update $\mathcal{B}(\mathbf{X}_t)$ so that $\mathcal{B}(\mathbf{X}_t) \subseteq \Theta$.

   (b) *Solution Sampling.* Use MIX-D (see Section 2.2) to sample ϖ solutions from $\mathcal{B}(\mathbf{X}_t)$. Aggregate the indices of the sampled solutions into a set $\mathcal{S}_t^{\mathrm{loc}}$.

   (c) *Selection of the Best.* Use SSM (see Section 2.3) to select the best solution whose index is $\hat{i}_t^{*\mathrm{loc}}$ from $\mathcal{S}_t^{\mathrm{loc}}$.

   (d) *Update the Best Solution.* $\mathbf{X}_{t+1} = \mathbf{x}_{\hat{i}_t^{*\mathrm{loc}}}$ and $t = t + 1$.

(3) *Termination.* Return $\mathbf{X}_t$.

Remark 2.2. The options for the stopping criterion are: (a) to perform HC once, (b) to perform HC until $\hat{i}_t^{*\mathrm{loc}} = \hat{i}_{t-1}^{*\mathrm{loc}}$, or (c) to perform HC until

$$\left|\bar{Y}_{\hat{i}_t^{*\mathrm{loc}}}\left(n_{\hat{i}_t^{*\mathrm{loc}}}\right) - \bar{Y}_{\hat{i}_{t-1}^{*\mathrm{loc}}}\left(n_{\hat{i}_{t-1}^{*\mathrm{loc}}}\right)\right| < \delta, \qquad (21)$$

where δ is an indifference-zone parameter. The motivation for (21) is similar to the rationale behind (20); with (21), however, HC stops if the perceived progress is too small.

Thus far, we have described how NP+SSM+HC works. In the next section, we establish its global convergence properties, which are not just an aggregation of the attributes of each component—partitioning, solution sampling, SSM, HC, and updating of the most-promising region—but also a result of their interactions.

3. PROPERTIES OF NP+SSM+HC

In Lemma 3.1, we establish that every solution in Θ is sampled infinitely often in the limit. The Strong Law of Large Numbers then leads us to almost sure convergence of NP+SSM+HC (see the Online Appendix for a proof).

LEMMA 3.1. *For all $i \in \{1, 2, \ldots, v\}$, $\lim_{k\to\infty} n_i(k) = \infty$.*

THEOREM 3.2. *NP+SSM+HC converges almost surely to one of the global optimal solutions as $k \to \infty$; that is, a solution $\mathbf{x}_{\hat{\imath}^*}$, where $\hat{\imath}^*$ is defined in* (4), *belongs to the set of optimal solutions almost surely as $k \to \infty$.*

Theorem 3.2 guarantees that our algorithm converges to a global optimal solution in the limit. In Theorem 3.3, Andradóttir [1999] provides us with an $O(k^{-1/2})$ convergence rate for the case of a unique optimum ($v^* = v$). Note that Theorem 3.2 holds when there are multiple optimal solutions, but in such cases we were unable to extend Andradóttir's convergence rate results to our algorithm. Theorem 3.3 also allows us to form a confidence interval for the maximum performance measure μ_v.

THEOREM 3.3. *If the optimal solution is unique, and $\hat{\imath}^*$ satisfies* (4), *then*

$$\sqrt{\Delta n_{free} k}\,(\bar{Y}_{\hat{\imath}^*}(n_{\hat{\imath}^*}(k)) - \mu_v) \;\Rightarrow\; N\left(0, \sigma_v^2\right),$$

where $\Rightarrow$ denotes convergence in distribution.

4. NUMERICAL EXPERIMENTS

We consider the performance of NP+SSM+HC relative to other optimization schemes on an (s, S) inventory problem and a three-stage buffer allocation problem. We will first describe the competing optimization schemes and characterize each test problem. Then we report selected results that emphasize key findings. In addition to NP+SSM+HC, the optimization approaches that we consider are:

NP. NP is our version of the algorithm, but not using SSM or HC. The algorithm takes Δn_{fixed} observations Y_{ip} from $\mathbf{x}_i$ on the first visit, and Δn_{free} additional observations on all other visits. NP selects the best solution over the set $\mathcal{S}_k$, $\hat{\mathbf{x}}^*(\mathcal{S}_k)$, as the one with the largest cumulative sample average, and it uses $\hat{\mathbf{x}}^*(\mathcal{S}_k)$ to determine the new most-promising region.

Random Search (*RS*) [*Andradóttir* 1996]. RS is a modified hill-climbing algorithm. Let $I_k \in \{1, 2, \ldots, v\}$ denote the index of the current solution on iteration k, and $C_i(k)$ denote the number of times $\mathbf{x}_i$ becomes the current solution up to iteration k, that is, $C_i(k) \equiv \sum_{p=0}^{k} \mathcal{I}\{I_p = i\}$, where $\mathcal{I}\{\nu\} = 1$ if ν is true and 0 otherwise. RS proceeds as follows:

(1) *Initialization.* Set $k = 0$, I_k to the index of a user-provided solution, $\mathbf{x}_0$ (If not given, use MIX-D to randomly sample a solution from Θ), and $C_i(k) = \mathcal{I}\{\mathbf{x}_i = \mathbf{x}_0\}, \forall i$.
(2) *Search.* Repeat Steps (2)(a)–(2)(c) until the simulation effort is exhausted:
    (a) Use MIX-D to uniformly sample a candidate solution $\mathbf{x}_{I'_k}$ over $\Theta \setminus \{I_k\}$, the entire feasible region except for solution I'_k.

(b) Take $\Delta n_{\text{fixed}} > 0$ observations of $Y_{I_k p}$ and $Y_{I'_k p}$, and compute the sample averages over these observations: $\bar{Y}_{I'_k}(\Delta n_{\text{fixed}})$ and $\bar{Y}_{I_k}(\Delta n_{\text{fixed}})$.

(c) Update I_k and $C_i(k)$:

$$I_{k+1} = \begin{cases} I'_k, & \text{if } \bar{Y}_{I'_k}(\Delta n_{\text{fixed}}) > \bar{Y}_{I_k}(\Delta n_{\text{fixed}}) \\ I_k, & \text{otherwise} \end{cases} \tag{22}$$

$$C_i(k+1) = C_i(k) + \mathcal{I}\{I_{k+1} = i\}$$
$$k = k+1.$$

(3) *Estimating the Optimal Solution.* The selected solution is $\mathbf{x}_{\hat{i}^*}$ where

$$\hat{i}^* = \arg\max_{1 \le i \le v} C_i(k). \tag{23}$$

Simulated Annealing (*SA*) [*Alrefaei and Andradóttir* 1999]. SA is almost identical to RS, but occasional downhill moves are allowed. Let the neighborhood structure be such that every solution is a neighbor of every other solution. The annealing temperature T is fixed. Equation (22) is replaced by

$$I_{k+1} \leftarrow \begin{cases} I'_k, & \text{if } U_k \le \exp\left\{-\Delta Y_k^+/T\right\} \\ I_k, & \text{otherwise.} \end{cases} \tag{24}$$

where $\Delta Y_k^+ = \max[\bar{Y}_{I_k}(\Delta n_{\text{fixed}}) - \bar{Y}_{I'_k}(\Delta n_{\text{fixed}}), 0]$, and $U_k \sim \text{Uniform}(0, 1)$. The selected solution is $\mathbf{x}_{\hat{i}^*}$ where

$$\hat{i}^* = \arg\max_{1 \le i \le v}\{\bar{Y}_i(n_i(k)) : n_i(k) > 0\}. \tag{25}$$

Remark 4.1. The past visit counts or observations are accumulated in RS and SA, respectively, for the purpose of estimating the optimal solution upon search termination (in (23) and (25)), but they are not used for the local comparison (in (22) or (24)).

We consider RS, SA and NP because they are globally convergent in our problem setting. We compare NP+SSM+HC to NP to study the role of selection-of-the-best schemes and HC. Each optimization scheme is given the same computational budget (the number of replications). Each algorithm is repeated for some number of times, and the results shown below are the *averaged* values across different searches.

Settings for the parameters of each of these search procedures (see Table II) were obtained by experimenting with a number of combinations of settings, and using ones that appear to yield good search performance. However, we did not formally conduct experiments to assess the optimal combination of these parameters for each procedure (Appendix C briefly explains how we obtain the annealing temperature T).

4.1 (*s*, *S*) Inventory Problem

In this classic problem [Koenig and Law 1985], the level of inventory of some discrete unit is periodically reviewed. Let $s \equiv x_1$ and $S \equiv x_2$. If the inventory position (units in inventory plus units on order minus units backordered) at a

Table II. Parameters used in the numerical experiments

Parameter	(s, S) inventory problem	Three-stage buffer allocation problem
Number of searches	1000	50
Superregion of R_k $\left(s(R_k)\right)$	$\Theta, \forall k$	$\Theta, \forall k$
Partitioning criterion	`BIGGEST RANGE`	`BIGGEST RANGE`
Number of subregions partitioned per R_k (ω)	2	2
Number of iterations without progress that triggers restart (k_0)	10	6
Number of solutions sampled from each subregion (ϑ)	3	5
Warm-up length of MIX-D and MIX-DS (length of Markov tour)	10	10
Minimum Number of observations taken from a sampled solution $\left(\Delta n_{\text{free}}\right)$	2	1
Number of observations for NP $\left(\Delta n_{\text{fixed}}\right)$	10	4
Indifference-zone parameter for SSM (δ)	1	0.5
First-stage number of observations for SSM (n_0)	10	4
Confidence level for SSM, $1 - \alpha$	0.9	0.9
Annealing temperature T	5	3
Initial solution	(70, 90)	(2, 2, 2, 2, 18)

review is found to be below x_1 units, then an order is placed to bring the inventory position up to x_2 units; otherwise, no order is placed. Demand is Poisson with mean rate of 25. The goal is to select x_i, $i = 1, 2$, such that the steady-state expected inventory cost per review period is minimized. The constraints on x_1 and x_2 are $x_1 - x_2 \leq 0,\ 20 \leq x_1 \leq 80,\ 40 \leq x_2 \leq 100,$ and $x_1, x_2 \in \mathcal{Z}^+$.

The number of feasible solutions is 2,901. The optimal inventory policy is (20, 53) with expected cost/period of 111.1265. The steady-state expected cost/period over the ranges of x_1 and x_2 considered (i.e., the response surface) is shown in Figure 4. To reduce the initial-condition bias, the average cost per period is computed after the first 100 review periods and averaged over the subsequent 30 periods. Other parameter settings are shown in Table II.

In this problem, the version of the NP+SSM+HC algorithm that we consider is NP+SSM(`REGION`), which is our algorithm without HC (skip Step (2)(d) of NP+SSM+HC algorithm and carry $\mathbf{x}_{\hat{i}_k^*} = \hat{\mathbf{x}}^*(\mathcal{S}_k)$ in to Step (2)(e). We do this to isolate the benefit of SSM. Figure 5 shows the steady-state expected average

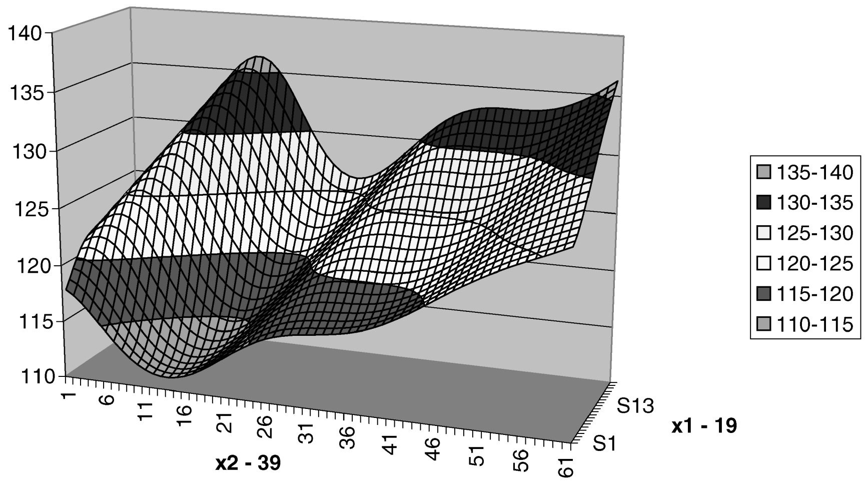

Fig. 4. Steady-state expected cost/period of the (s, S) inventory problem.

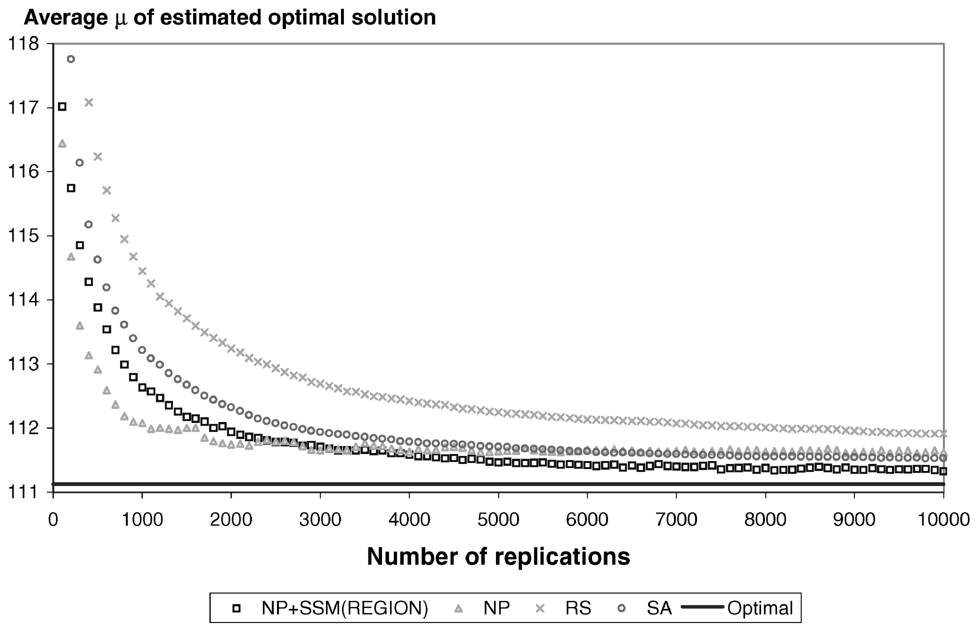

Fig. 5. Steady-state expected cost/period of the current optimal-solution estimates $\mu_{\hat{i}^*}$ at each point in time for the (s, S) inventory problem.

cost per period of the *current* optimal-solution estimates $\mu_{\hat{i}^*}$ (depending on the algorithm, $\hat{i}^*$ is defined in (4), (23), or (25)) at each point in time averaged over 1000 searches. Initially, the performance of NP is better than other optimization methods. However, as the simulation effort (i.e., the number of simulation replications) increases, NP+SSM(REGION) outperforms the rest despite the small number of solutions seen by NP+SSM(REGION) relative to other optimization algorithms (see Figure 6). This illustrates the dilemma of selection versus

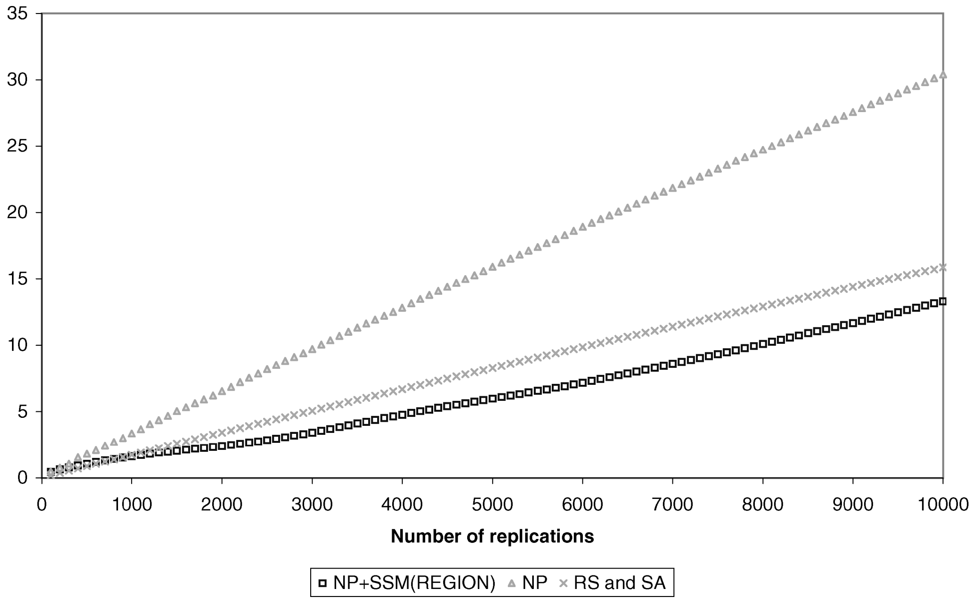

Fig. 6. Percentage of solutions seen by the optimization schemes when applied to the (s, S) inventory problem.

exploration. When the available simulation effort is small, NP+SSM(REGION) is unable to see much of Θ because it exhausts more simulation effort per search iteration than NP does (recall that NP+SSM(REGION) uses SSM(REGION) on every iteration, whereas NP does not use SSM at all). However, as the optimization progresses, the benefit of successively making good selections on every iteration finally pays off, and the performance of NP+SSM(REGION) surpasses that of other optimization schemes.

4.2 Three-Stage Buffer Allocation Problem

We consider a three-stage flow line with finite buffer storage space in front of stations 2 and 3 (denoted by x_4 and x_5, respectively) and an infinite number of jobs in front of station 1. There is a single server at each station, and the service time at station h is exponentially distributed with service rate $x_h, h = 1, 2, 3$. If the buffer of station h is full, then station $h - 1$ is blocked and a finished job cannot be released from station $h-1$. The total buffer space and the service rates are limited. The goal is to find a buffer allocation and service rates such that the throughput (average output of the flow line per unit time) is maximized. We obtained the balance equations for the underlying Markov chain from Buzacott and Shantikumar [1993]. The constraints (in the form of (2)) are: $x_1 + x_2 + x_3 \leq 20; x_4 + x_5 \leq 20; -x_4 - x_5 \leq -20; 1 \leq x_h \leq 20, h = 1, 2, \ldots, 5; x_h \in \mathcal{Z}^+$.

The number of feasible solutions is 21,660. The optimal solutions are $(6, 7, 7, 12, 8)$ and $(7, 7, 6, 8, 12)$ with an expected throughput of 5.776. In the simulation, the throughput is estimated after the first 2000 units have been produced, and it is averaged over the subsequent 50 units released. When HC is used, it is

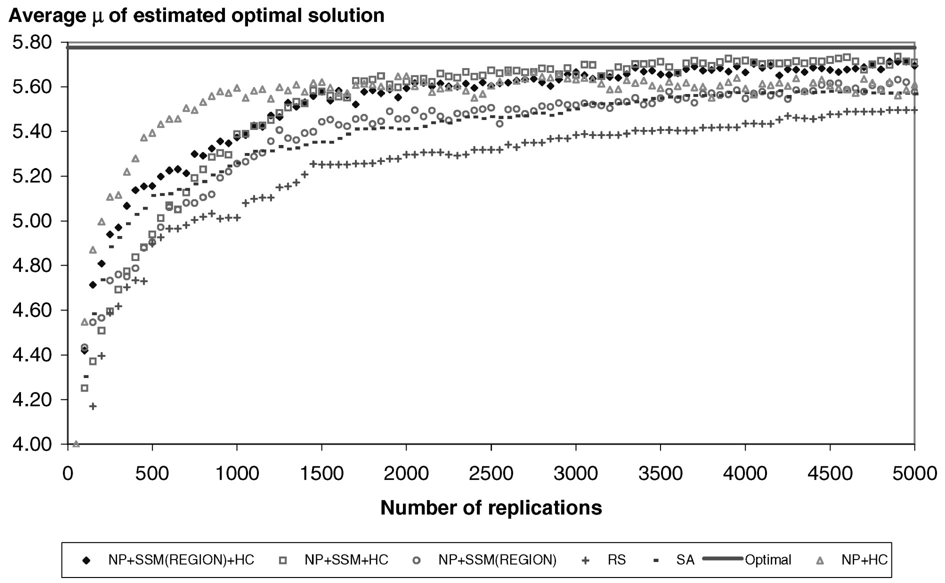

Fig. 7. Expected throughput of the current optimal-solution estimate at each point in time for the buffer allocation problem.

Table III. Fraction of the total number of solution that the optimization scheme has seen when the search terminates

Algorithm	% of number of solution seen
NP+SSM(REGION)+HC	1.64
NP+SSM+HC	1.41
NP+SSM(REGION)	1.80
RS	2.85
SA	2.84

performed on all iterations; HC stops when $|\bar{Y}_{\hat{i}_t^*}(n_{\hat{i}_t^*}) - \bar{Y}_{\hat{i}_{t-1}^*}(n_{\hat{i}_{t-1}^*})| < \delta$ where $\mathbf{x}_{\hat{i}_t^*}$ is the current best solution on HC iteration t, and δ is an indifference-zone parameter for SSM (i.e., the amount that a user cares to distinguish between solutions; see (15)); the number of candidate solutions on each HC step is 3, and the neighborhood step size is 1. Other parameter values are shown in Table II. To isolate the impact of SSM, we also consider an NP+HC search where the best solution from the NP method is the starting solution for the HC search (see Section 2.4). The HC parameters we use in NP+HC are identical to what we use for NP+SSM+HC searches.

Figure 7 shows the expected throughput of the current optimal-solution estimate $\mu_{\hat{i}^*}$ averaged over 50 searches at each point in time. Initially, the NP+HC search performs well because this test problem has low variability, large feasible space (see Table III for the fraction of the total number of solutions that an optimization scheme has visited when the search terminates) and relatively regular response surface. Such characteristics are advantageous for an optimization scheme such as NP+HC, which visits many solutions quickly and does not spend much time on estimating the performance measure. Nevertheless, the ability to correctly select good solutions eventually matters; Figure 7 shows that as the simulation effort increases, NP+SSM(REGION)+HC and NP+SSM+HC outperform the rest.

The benefit of HC is manifested through the favorable performance of NP+SSM(REGION)+HC compared to that of NP+SSM(REGION). This result confirms our conjecture that HC is advantageous for problems with large $|\Theta|$ and clustered good solutions; this three-stage buffer allocation problem has $|\Theta| = 21,660$ with 5 decision variables, and good solutions are in close proximity.

Next, we examine the value of option REGION. NP+SSM(REGION)+HC noticeably outperforms NP+SSM+HC in the initial phase of the search. Option REGION is helpful when the simulation budget is limited because SSM(REGION) consumes less simulation effort per search iteration than SSM does (recall that SSM(REGION) stops when all surviving solutions belong to the same subregion); therefore, NP+SSM(REGION)+HC is able to see more solutions than NP+SSM+HC does, and thus NP+SSM(REGION)+HC is more likely to find good solutions given limited simulation effort.

In addition to the two test problems above, we have also experimented with a known response function with added white noise whose variance is inversely proportional to the value of function (see the Zimmermann's function section in the Online Appendix). This example allows us to examine the performance of the algorithms when there is significant variability near the optimal solution.

Unlike the previous two problems, NP+SSM and NP+SSM(REGION)+HC outperform all other optimization schemes at all levels of simulation effort. This result illustrates the significance of correct selection on the performance of the optimization schemes when variability is high. Although the underlying response surface is smooth and monotonic, this problem is difficult because the noise is not only large, but it also becomes larger in the vicinity of the optimal solution. For such problems, it is critical to make a good selection on every iteration.

5. CONCLUSION

We have proposed an optimization-via-simulation algorithm with the goal of establishing both provable convergence and good empirical performance. NP+SSM+HC consists of a global guidance system, selection of the best, and local improvement. We use the NP method as our global guidance system to ensure that the search not only advances toward optimal solutions, but it also reaches one of them, if there is enough simulation effort. While the NP method gives us the convergence guarantee, SSM enhances the performance of the NP method by controlling the selection error, and HC improves it further by intensifying the search near good solutions. This is critical because the convergence guarantee comes only as each feasible solution is visited infinitely often, while in practice some solutions may not be visited at all if the feasible space is large. Notice that we see good performance in our examples even though only a small fraction of solutions are simulated.

Table IV. Notation for k_0 calculation.

Notation	Definition
p_0	probability of encountering solutions better than $\mathbf{x}_{i_0}$ $\equiv \frac{\left|\{\mathbf{x}_i : \mu_i \geq \mu_{i_0}\}\right|}{v-1}$, where $\lvert A\rvert$ denotes the cardinality of a set A
H_0	null hypothesis: $p_0 \geq \beta$
v	number of feasible solutions in Θ
ϑ	number of sampled solutions per subregion
i_0	index of the current best solution
β	threshold probability of restarting
k_0	threshold number of iterations to restart
α	Type-I error in rejecting H_0 when it is true

The motivation behind NP+SSM+HC is to make optimization-via-simulation algorithms adapt to variability and to characteristics of the response surface. Our algorithms show promise in numerical tests (see also the extensive study in Pichitlamken [2002]). Still, there is substantial room to be even more adaptive. A key contribution of our approach is that it provides a flexible framework within which many sensible schemes can be embedded without sacrificing provable performance. For example, in the selection of the best, SSM can be replaced by an extension of Rinott's procedure in Boesel et al. [2003b]; in the local search, the hill-climbing algorithm can be augmented by tabu search, simulated annealing or a genetic algorithm.

NP+SSM+HC was designed for problems with integer-valued decision variables, and it will work best in that environment. Continuous-valued decision variables can, of course, be discretized, so NP+SSM+HC could be adapted to continuous-decision-variable problems in that (perhaps very inefficient) way. If the convergence guarantee is sacrificed, then NP+SSM+HC will work directly with continuous-valued decision variables by replacing MIX-D and MIX-DS with one of the many algorithms for sampling from a continuous feasible region.

APPENDIXES

A. DETERMINING THE RESTARTING THRESHOLD

We consider restarting NP+SSM+HC when the current most promising region has been the same singleton for k_0 iterations. We restart when the estimated probability of encountering solutions better than the current best solution, say $\mathbf{x}_{i_0}$, is low. The notation is defined in Table IV.

If the null hypothesis H_0 is not rejected, the search continues with the current branching scheme. On the other hand, if H_0 is rejected, then the search restarts. The test is based on the number of consecutive iterations on which we do not find a solution better than $\mathbf{x}_{i_0}$. To control Type-I error, we determine a threshold number of consecutive NP iterations, k_0, in which $\mathbf{x}_{i_0}$ remains the most promising. The threshold k_0 can be considered as the number of Bernoulli trials without success in rejecting H_0, where the probability of success is β.

Therefore, we have that

$$\begin{aligned}\alpha &\geq \max_{\beta \leq p_0 \leq 1} \Pr\{\text{number of consecutive iterations without restart} = k_0 \mid H_0 \\ &\qquad \text{is true}\} \\ &= \max_{\beta \leq p_0 \leq 1} (1 - p_0)^{\vartheta k_0} \\ &\geq (1 - \beta)^{\vartheta k_0}\end{aligned}$$

Thus, $k_0 = (\ln \alpha)/(\vartheta \ln(1 - \beta))$. We use $\alpha = 0.04$ and $\beta = 0.1$ in the numerical experiments (Section 4). Notice that the analysis above is based on two approximations:

(1) Sampling is independent and uniform *with* replacement, and p_0 does not change: However, we actually sample *without* replacements, that is, we obtain distinct solutions. Nevertheless, this discrepancy is insignificant if the size of Θ is large relative to number of solutions we sample. In addition, the asymptotic uniformity of MIX-D and MIX-DS also helps alleviate the difference.

(2) If a solution that is better than $\mathbf{x}_{i_0}$ exists in the pool of sampled solutions, then it is selected. Of course, there will sometimes be selection error, but this error is controlled by SSM.

The restarting threshold k_0 could also be used as a stopping criterion for NP+SSM+HC, that is, the search terminates if some singleton region remains the most promising for k_0 consecutive iterations.

B. ALGORITHM MIX-DS

We use MIX-DS to sample solutions from the surrounding region $\Theta \setminus R_k$.

To make the notation more compact, we can express the constraints (5) in vector form. First, we index the constraints, other than integrality, as follows:

$$\sum_{i=1}^{q} \tilde{a}_{ij}^{\tau} x_i \leq \tilde{b}_j^{\tau}, \quad j = 1, 2, \ldots, p^{\tau} + 2q$$

where

$$\tilde{a}_{ij}^{\tau} = \begin{cases} a_{ij}^{\tau}, & j = 1, 2, \ldots, p^{\tau} \\ -\mathcal{I}\{i = (j - p^{\tau})\}, & j = p^{\tau} + 1, p^{\tau} + 2, \ldots, p^{\tau} + q \\ \mathcal{I}\{i = (j - p^{\tau} - q)\}, & j = p^{\tau} + q + 1, p^{\tau} + q + 2, \ldots, p^{\tau} + 2q \end{cases}$$

and

$$\tilde{b}_j^{\tau} = \begin{cases} b_j^{\tau}, & j = 1, 2, \ldots, p^{\tau} \\ -l_{j-p^{\tau}}^{\tau}, & j = p^{\tau} + 1, p^{\tau} + 2, \ldots, p^{\tau} + q \\ u_{j-p^{\tau}-q}^{\tau}, & j = p^{\tau} + q + 1, p^{\tau} + q + 2, \ldots, p^{\tau} + 2q. \end{cases}$$

The indicator function $\mathcal{I}\{\epsilon\}$ is one if ϵ is true and zero if ϵ is false. Then, excluding integrality, constraint (5) becomes

$$\tilde{\mathbf{a}}_j^{\tau} \mathbf{x} \leq \tilde{b}_j^{\tau}, \quad j = 1, 2, \ldots, p^{\tau} + 2q, \tag{26}$$

where $\tilde{\mathbf{a}}_j^{\tau}$ is a row vector: $(\tilde{a}_{1j}^{\tau}, \tilde{a}_{2j}^{\tau}, \ldots, \tilde{a}_{qj}^{\tau})$.

Algorithm MIX-DS

Input: A closed convex region τ of the form (5) which is a proper subset of another closed convex region Θ of the form (2), number of solutions to sample ϑ, length of warm-up period T, and a starting solution $\mathbf{x}_0 \notin \tau$ but $\mathbf{x}_0 \in \Theta$.
Output: ϑ random integer solutions that are in $\Theta \setminus \tau$.
Procedure:

(1) *Initialization.* Set $\mathbf{X}_0 = \mathbf{x}_0$, $t = 1$, and the set of sampled solution $\mathcal{S} = \emptyset$.
(2) *Generating Random Solutions.*
   (a) Sample a random *scaled* direction $\mathbf{D}_t$ (see Step (2)(a) of MIX-D for details).
   (b) Determine the set of possible integer increments within Θ, $\mathcal{C}(\mathbf{X}_{t-1}, \mathbf{D}_t, \Theta)$ (see Step (2)(b) of MIX-D).
   (c) Divide the constraints of τ (see (26)) into two sets: those which $\mathbf{X}_{t-1}$ has satisfied (denoted as $I_t^y(\tau)$) and those which $\mathbf{X}_{t-1}$ has not satisfied (denoted as $I_t^n(\tau)$):

$$I_t^y(\tau) = \left\{ j : \tilde{\mathbf{a}}_j^\tau \mathbf{X}_{t-1} \le \tilde{b}_j^\tau,\ \ 1 \le j \le p^\tau + 2q \right\} \tag{27}$$
$$\begin{aligned} I_t^n(\tau) &= \left\{ j : \tilde{\mathbf{a}}_j^\tau \mathbf{X}_{t-1} > \tilde{b}_j^\tau,\ \ 1 \le j \le p^\tau + 2q \right\} \\ &= \left\{ 1, 2, \ldots, p^\tau + 2q \right\} \setminus I_t^y(\tau). \end{aligned} \tag{28}$$

   (d) Determine if a move in direction $\mathbf{D}_t$ passes through τ:
      (i) Compute the minimum increment $\underline{C}_t(\tau)$ that brings the tour from $\mathbf{X}_{t-1}$ into τ if it is possible at all in the direction $\mathbf{D}_t$, that is, $\underline{C}_t(\tau) \ge 0$ such that $(\mathbf{X}_{t-1} + (\underline{C}_t(\tau) - 1) \cdot \mathbf{D}_t) \notin \tau$, but $(\mathbf{X}_{t-1} + \underline{C}_t(\tau) \cdot \mathbf{D}_t) \in \tau$. The increment $\underline{C}_t(\tau)$ is determined by:

$$\underline{C}_{jt}(\tau) = \begin{cases} \left\lceil \frac{\tilde{b}_j^\tau - \tilde{\mathbf{a}}_j^\tau \mathbf{X}_{t-1}}{\tilde{\mathbf{a}}_j^\tau \mathbf{D}_t} \right\rceil, & \text{if } \tilde{\mathbf{a}}_j^\tau \mathbf{D}_t < 0 \\ \infty, & \text{otherwise} \end{cases} \tag{29}$$
$$\underline{C}_t(\tau) = \max_{j \in I_t^n(\tau)} \underline{C}_{jt}(\tau). \tag{30}$$

      If $\underline{C}_t(\tau) = \infty$, then a move in direction D_t will definitely not pass through τ; go to Step (2)(d)(iv). However, if $\underline{C}_t(\tau) < \infty$, it is possible that D_t will pass through τ in positive multiples of D_t.
      (ii) Compute the maximum increment $\bar{C}_t(\tau)$ that brings the random tour from $\mathbf{X}_{t-1}$ into τ but not out of τ if it is possible at all in direction $\mathbf{D}_t$, that is, $\bar{C}_t(\tau) \ge 0$ such that $(\mathbf{X}_{t-1} + \bar{C}_t(\tau) \cdot \mathbf{D}_t) \in \tau$, but $(\mathbf{X}_{t-1} + (\bar{C}_t(\tau) + 1) \cdot \mathbf{D}_t) \notin \tau$. The increment $\bar{C}_t(\tau)$ is determined by:

$$\bar{C}_{jt}(\tau) = \begin{cases} \left\lfloor \frac{\tilde{b}_j^\tau - \tilde{\mathbf{a}}_j^\tau \mathbf{X}_{t-1}}{\tilde{\mathbf{a}}_j^\tau \mathbf{D}_t} \right\rfloor, & \text{if } \tilde{\mathbf{a}}_j^\tau \mathbf{D}_t > 0 \\ \infty, & \text{otherwise} \end{cases} \tag{31}$$
$$\bar{C}_t(\tau) = \min_{j \in I_t^y(\tau)} \bar{C}_{jt}(\tau). \tag{32}$$

      (iii) If $\underline{C}_t(\tau) > \bar{C}_t(\tau)$, a move in *positive* multiples of $\mathbf{D}_t$ cannot bring the random tour into the inner region τ, go to Step (2)(d)(iv). On the other hand, for $C \in [\underline{C}_t(\tau), \bar{C}_t(\tau)]$, we have that $(\mathbf{X}_{t-1} + C\mathbf{D}_t) \in \tau$. Thus, the set of possible increments that keep the tour outside τ while remaining inside Θ is

$$\mathcal{C}(\mathbf{X}_{t-1}, \mathbf{D}_t, \Theta) \leftarrow \mathcal{C}(\mathbf{X}_{t-1}, \mathbf{D}_t, \Theta) \setminus [\underline{C}_t(\tau), \bar{C}_t(\tau)]. \tag{33}$$

      If $\mathcal{C}(\mathbf{X}_{t-1}, \mathbf{D}_t, \Theta) = \emptyset$, go to Step (2)(a); otherwise, go to Step (2)(e).
      (iv) Check if negative increments can bring the random tour into the region τ:
         (A) Reverse the direction $\mathbf{D}_t$ to get $-\mathbf{D}_t$, that is, $-D_{it} = (-1) \cdot D_{it}$ for $i = 1, 2, \ldots, q$.
         (B) Substitute $-\mathbf{D}_t$ for $\mathbf{D}_t$, and use Eqs. (29) and (30) to get $\bar{C}_t(\tau)$. Suppose C is the result of Eqs. (29) and (30), then $\bar{C}_t(\tau) \leftarrow (-1) \cdot C$. If $\bar{C}_t(\tau) = -\infty$, then go to Step (2)(e).

(C) Compute $\underline{C}_t(\tau)$ with $-\mathbf{D}_t$ being substituted for $\mathbf{D}_t$ in Eqs. (31) and (32), and then reverse the sign of the result, that is, let C be the result of Eqs. (31) and (32) with $-\mathbf{D}_t$, then $\underline{C}_t(\tau) \leftarrow (-1) \cdot C$.

(D) If $\underline{C}_t(\tau) \leq \bar{C}_t(\tau)$, then update the set of possible increments with (33).

(e) Sample C_t uniformly over the set $\mathcal{C}(\mathbf{X}_{t-1}, \mathbf{D}_t, \Theta)$.

(f) Update $\mathbf{X}_t$ and t;

$$\begin{aligned} \mathbf{X}_t &= \mathbf{X}_{t-1} + C_t \mathbf{D}_t \qquad (34) \\ t &= t + 1. \end{aligned}$$

(g) If $t < T$, go to Step (2)(a); otherwise, insert $\mathbf{X}_T$ into the set $\mathcal{S}$ if $\mathbf{X}_T \notin \mathcal{S}$. If $\mathcal{S}$ has ϑ solutions, return the set $\mathcal{S}$; otherwise, set $t = 0$ and $\mathbf{X}_0 = \mathbf{X}_T$ and go to Step (2)(a).

Remark B.1. The left-hand sides of Eqs. (29)–(32) are nonnegative.

Notice that Algorithm MIX-DS is essentially Algorithm MIX-D over the feasible space Θ, but not all increments C_t are permitted. Similar to MIX-D, the stochastic process induced by MIX-DS is a stationary and irreducible Markov chain over a discrete and finite space $\Theta \setminus \tau$. Theorem B.2 below shows that this Markov chain also has a uniform limiting distribution (see the Online Appendix for the proof).

Theorem B.2. *The limiting distribution of the Markov chain induced by MIX-DS* $\{\mathbf{X}_t : t \geq 0\}$ *is uniform.*

C. DETERMINING THE ANNEALING TEMPERATURE

We determined the annealing temperature T for SA via a mix of calculation and experimentation. In Eq. (24), we arbitrarily set the threshold acceptance probability to be 0.7 and set ΔY_k^+ to be the indifference-zone parameter δ. Then we solved for the annealing temperature T as $T = -\delta/\ln(0.7)$. We rounded this number up, and increased it further if it seemed to help in experiments. For example, for the (s, S) inventory problem, $T = -\delta/\ln(0.7) \approx 2.8$, but we used $T = 5$.

ELECTRONIC APPENDIX

The electronic appendix for this article can be accessed in the ACM Digital Library.

ACKNOWLEDGMENTS

We would like to thank General Motors R&D Center, and specifically Dr. Jeffrey D. Tew, for support of this work, and also Dr. Sigrún Andradóttir of Georgia Tech for her ideas and assistance. We also acknowledge the guidance of the co-Editor Michael Fu and two referees on revising the presentation of the article.

REFERENCES

Alrefaei, M. H. and Andradóttir, S. 1999. A simulated annealing algorithm with constant temperature for discrete stochastic optimization. *Manage. Sci. 45*, 748–764.

Andradóttir, S. 1995. A method for discrete stochastic optimization. *Manage. Sci. 41*, 1946–1961.

ANDRADÓTTIR, S. 1996. A global search method for discrete stochastic optimization. *SIAM J. Optim. 6*, 513–530.

ANDRADÓTTIR, S. 1998. Simulation optimization. In *Handbook of Simulation*. J. Banks, Ed. Wiley, New York.

ANDRADÓTTIR, S. 1999. Accelerating the convergence of random search methods for discrete stochastic optimization. *ACM Trans. Model. Comput. Simulat. 9*, 349–380.

BOESEL, J. 1999. Search and selection for large-scale stochastic optimization. Ph.D. Dissertation, Department of Industrial Engineering and Management Sciences, Northwestern Univ., Evanston, Ill.

BOESEL, J., NELSON, B. L., AND ISHII, N. 2003a. A framework for simulation-optimization software. *IIE Trans. 35*, 3, 221–230.

BOESEL J., NELSON, B. L., AND KIM, S.-H. 2003b. Using ranking and selection to 'clean up' after simulation optimization. *Oper. Res.*, forthcoming.

BUZACOTT, J. A. AND SHANTIKUMAR, J. G. 1993. *Stochastic Models of Manufacturing Systems*. Prentice-Hall, Englewood Cliffs, N.J.

FU, M. C. 1994. Optimization via simulation: A review. *Ann. Oper. Res. 53*, 199–247.

FU, M. C. 2002. Optimization for simulation: Theory vs. practice. *INFORMS J. Comput. 14*, 192–215.

GELFAND, S. B. AND MITTER, S. K. 1989. Simulated annealing with noisy or imprecise energy measurements. *J. Optim. Theory Appl. 62*, 49–62.

GOLDSMAN, D. AND NELSON, B. L. 1998. Comparing systems via simulation. In *Handbook of Simulation.* J. Banks, Ed. Wiley, New York.

GUTJAHR, W. J. AND PFLUG, G. CH. 1996. Simulated annealing for noisy cost functions. *J. Glob. Optimiz. 8*, 1–13.

JACOBSON, S. H. AND SCHRUBEN, L. W. 1989. Techniques for simulation response optimization. *Oper. Res. Lett. 8*, 1–9.

KOENIG, L. W. AND LAW, A. M. 1985. A procedure for selecting a subset of size m containing the l best of k independent normal populations, with applications to simulation. *Communi. Stat.: Simul. Comput. 14*, 719–734.

PICHITLAMKEN, J. 2002. A combined procedure for optimization via simulation. Ph.D. Dissertation, Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, Illinois.

PICHITLAMKEN, J. AND NELSON, B. L. 2001. Selection-of-the-best procedures for optimization via simulation. In *Proceedings of the 2001 Winter Simulation Conference* (Arlington, Va, Dec.) B. A. Peters, J. S. Smith, D. J. Medeiros, and M. W. Rohrer, Eds. IEEE. Piscataway, N.J., 401–407.

SHI, L. AND CHEN, C.-H. 2000. A new algorithm for stochastic discrete resource allocation optimization. *J. Disc. Event Dyn. Syst. 10*, 271–294.

SHI, L. AND ÓLAFSSON, S. 2000. Nested partitions method for stochastic optimization. *Meth. Comput. Appl. Prob. 2*, 271–291.

SHI, L., ÓLAFSSON, S., AND CHEN, Q. 2001. An optimization framework for product design. *Manage. Sci. 47*, 1681–1692.

SMITH R. L. 1984. Efficient Monte Carlo procedures for generating points uniformly distributed over bounded regions. *Oper. Res. 32*, 1296–1308.

SWISHER, J. R., HYDEN, P. D., JACOBSON, S. H., AND SCHRUBEN, L. W. 2004. A survey of recent advances in discrete input parameter discrete-event simulation optimization. *IIE Trans.*, forthcoming.

WOLSEY, L. A. 1998. *Integer Programming*. Wiley, New York.

YAN, D. AND MUKAI, H. 1992. Stochastic Discrete Optimization. *SIAM J. Cont. Optim. 30*, 594–612.

Received May 2002; revised September 2002 and February 2003; accepted March 2003