Integer-Ordered Simulation Optimization using R-SPLINE: Retrospective Search with Piecewise-Linear Interpolation and Neighborhood Enumeration

HONGGANG WANG, Rutgers University
RAGHU PASUPATHY, Virginia Tech and IBM T. J. Watson Research
BRUCE W. SCHMEISER, Purdue University

We consider simulation-optimization (SO) models where the decision variables are integer ordered and the objective function is defined implicitly via a simulation oracle, which for any feasible solution can be called to compute a point estimate of the objective-function value. We develop R-SPLINE—a Retrospective-search algorithm that alternates between a continuous Search using Piecewise-Linear Interpolation and a discrete Neighborhood Enumeration, to asymptotically identify a local minimum. R-SPLINE appears to be among the first few gradient-based search algorithms tailored for solving integer-ordered local SO problems. In addition to proving the almost-sure convergence of R-SPLINE's iterates to the set of local minima, we demonstrate that the probability of R-SPLINE returning a solution outside the set of true local minima decays exponentially in a certain precise sense. R-SPLINE, with no parameter tuning, compares favorably with popular existing algorithms.

Categories and Subject Descriptors: G.1.6 [**Numerical Analysis**]: Optimization—*Unconstrained optimazation, integer programming, gradient methods*; G.3 [**Probability and Statistics**]: *Probabilistic algorithms (including Monte Carlo)*; G.4 [**Mathematical Software**]: *Algorithm design and analysis*; I.6.1 [**Simulation and Modeling**]: Simulation Theory

General Terms: Algorithms, Design, Theory

ACM Reference Format:
Wang, H., Pasupathy, R., and Schmeiser, B. W. 2013. Integer-ordered simulation optimization using R-SPLINE: Retrospective Search with Piecewise-Linear Interpolation and Neighborhood Enumeration. *ACM Trans. Model. Comput. Simul.* 23, 3, Article 17 (July 2013), 24 pages.
DOI: http://dx.doi.org/10.1145/2499913.2499916

1. INTRODUCTION

We consider stochastic simulation-optimization (SO) problems—optimization problems where the real-valued objective function g is defined by a black-box Monte Carlo simulation oracle parameterized by the decision variables x. The function g is implicit in the oracle in that at any design point x, only a point estimate of the objective function value $g(x)$ can be obtained (in finite time) by calling the oracle. The oracle is consistent in that the actual value $g(x)$ is available in an asymptotic sense, for example, by averaging (or other calculation) values obtained from an infinite number of oracle calls.

SO problems have recently grown in popularity because of the flexibility that the problem formulation naturally affords. Specifically, since SO problems allow posing

Early ideas contained in this article appeared in Wang and Schmeiser [2008] and Wang [2009].
Later development was supported in part by Office of Naval Research contract N000141110419.
Author's address: R. Pasupathy; email: pasupath@vt.edu.

DOI: http://dx.doi.org/10.1145/2499913.2499916

Table I. Classification of SO Algorithms

	Finite (General)	Integer Ordered (Finite)	Integer Ordered (Infinite)
Local	NA	Coordinate Search [Hong 2005], COMPASS [Hong and Nelson 2006], DSA [Lim 2012]	Coordinate Search [Hong 2005], COMPASS [Hong and Nelson 2006], DSA [Lim 2012]
Global	Too numerous. (See Ranking & Selection Methods [Kim and Nelson 2006].)	Stochastic Ruler Methods [Alrefaei and Andradóttir 2001; Yan and Mukai 1992], Random Search Methods [Andradóttir 1995, 1996], Nested Partition Methods [Pichitlamken and Nelson 2003; Shi and Ólafsson 2000], Stochastic Comparison [Gong et al. 1999]	Modified Stochastic Ruler [Alrefaei and Andradóttir 2001], Random Search Methods [Andradóttir 1995]

A classification of discrete SO problems by nature of the feasible solution space and nature of the requested solution, along with representative (provably convergent) methods for solution. The problem considered in this article falls in the two rightmost cells of the top row.

an optimization problem implicitly through a stochastic simulation, any level of model complexity can be incorporated within the problem, at least in principle. Implementers in a wide variety of real-world settings (e.g., vehicular traffic systems, production and logistics, telecommunications) thus find SO formulations to be particularly expedient within their stochastic optimization settings. See Andradóttir [2006], Barton and Meckesheimer [2006], Fu [2002], and Spall [2003] for application and methodological overviews on the subject.

Our interest is limited to SO problems seeking *locally optimal solutions* (defined in Section 2.1), and having integer-ordered decision variables x. Such applications arise, for example, when the decision variables are inventory reorder points and quantities, numbers of machines, numbers of stock options, or staffing levels. Within the world of SO problems, integer-ordered SO problems seeking local optima seem to be common.

The popularity of local SO problems on integer-ordered spaces is evidenced by the types of problem entries in the library of SO problems [Pasupathy and Henderson 2006, 2011]. (This library of SO problems is constructed using problems selected from the open literature and from voluntary submissions.) Of the 50 total problems included within the library, 25 are SO problems on integer-ordered spaces and seeking a local minimum. Notwithstanding the pervasiveness of local SO problems on integer-ordered spaces, general algorithms tailored for this class are somewhat lacking. Table I, a fairly representative list of available algorithms for solving SO problems on discrete spaces, illustrates this fact. There is a clear proliferation of algorithms to solve global SO problems, with a large proportion of those focused on finite spaces. (Since the literature for Ranking and Selection is too voluminous for exhaustive listing, we have listed only a review article, Kim and Nelson [2006].) For the context of local SO problems, coordinate search [Hong 2005], COMPASS [Hong and Nelson 2006], and Discrete Stochastic Approximation [Lim 2012] (henceforth DSA) are notable examples.

The goal, of course, is to obtain a high-quality local minimum. R-SPLINE is designed to quickly find a high-quality minimum, although the convergence guarantee is only for a local minimum. The acronym R-SPLINE describes its algorithmic structure. The R stands for *Retrospective*, an algorithm framework in which a sequence of sample path problems are solved, each with larger sample size, using the previous solution for a warm start. For each sample size, the SPLINE algorithm is applied. SPLINE is composed of two parts: a continuous Search with Piecewise-Linear Interpolation (SPLI) and a discrete Neighborhood Enumeration (NE). SPLI allows large non-coordinate-direction steps to better solutions using gradient-search logic; NE checks whether the result of SPLI is locally optimal and moves to the best solution of the neighborhood.

SPLI and NE alternate until a local solution to the current sample-path problem is found. The Retrospective framework then increases the sample size to obtain a new sample-path problem. Within each Retrospective iteration, the same sample size is used for all solution evaluations, making the use of common random numbers natural.

The component ideas of R-SPLINE are not new. The retrospective framework, with roots in Rubinstein and Shapiro [1993] and Healy and Schruben [1991] was introduced in the context of stochastic root-finding problems [Chen 1994; Chen and Schmeiser 1994, 2001; Pasupathy 2010; Pasupathy and Kim 2011; Pasupathy and Schmeiser 2009] and subsequently used for SO problems on continuous spaces [Jin 1998; Kim et al. 2012; Pasupathy 2010]. The use of phantom gradients, as in SPLI, also has appeared in various forms and contexts before. The earliest example of such use within the SO setting seems to be by Whitney et al. [2001], where the simultaneous-perturbation stochastic approximation (SPSA) [Spall 2003] is extended to discrete sets. (No convergence guarantees are provided for the resulting algorithm.) More recently, phantom gradients have appeared in the context of cutting-plane algorithms for the call-center staffing problem [Atlason et al. 2008] and model-based sequential approximation algorithms [Deng and Ferris 2006, 2009]. Despite the similarities, these two algorithms are not competitors because the former is for a specific application context, while the latter is aimed at nondifferentiable global SO problems.

1.1. Competitors

The primary competitors for R-SPLINE are COMPASS [Hong and Nelson 2006, 2007], its recently enhanced version within Industrial Strength COMPASS (ISC) [Hong et al. 2010; Xu et al. 2010], and ISC with the Adaptive Hyperbox Algorithm (ISC-AHA) [Xu et al. 2011], an enhancement for use with problems having dimension greater than ten. COMPASS was originally designed to solve local SO problems on integer-ordered spaces. Like most other SO algorithms, COMPASS consists of repeating steps that strategically explore the search space while estimating (to different extents) the objective function value at the visited solutions. The algorithm's novelty rests on the notion of the "most promising region" (MPR), defined as the set of feasible solutions that are closer (in Euclidean norm) to the best estimated solution than to any other observed solution. COMPASS evolves by updating and uniformly searching for new solutions within the MPR, and simultaneously increasing the precision of the estimated objective function value at each of the already visited solutions.

ISC extends COMPASS towards providing the user with solutions that are in a sense better than just providing a local minimum. ISC's explicit intent is returning a high-quality local optimum with probabilistic guarantees at termination. This is accomplished by including a preliminary global phase (Phase I) that feeds a population of good starting solutions to COMPASS. COMPASS in turn works from each of the starting solutions to attain a set of local minima (Phase II). ISC ends with a clean-up phase (Phase III) that evaluates each of the local solutions attained by COMPASS to identify and return the best among the identified local minima.

ISC has recently received much attention and is arguably the most efficient among algorithms (for local SO on integer-ordered spaces) capable of providing asymptotic or rigorous finite-time probabilistic guarantees. To this extent, virtually all assessment of R-SPLINE will be based on a comparison with ISC and its high-dimensional enhancement ISC-AHA. R-SPLINE, like COMPASS, is designed to rapidly converge to local extrema on integer-ordered SO problems. Unlike COMPASS, however, R-SPLINE relies on phantom gradients to guide searching within the feasible region. This, in combination with the strategic sampling framework within which R-SPLINE lives, gives R-SPLINE a crucial advantage when identifying local extrema. As we demonstrate through numerical experiments, this issue manifests as R-SPLINE reaching the

vicinity of the local minimum faster than ISC. This difference in convergence rate between R-SPLINE and ISC tends to become even more pronounced in high-dimensional problems. (It is worth emphasizing that since R-SPLINE is a local algorithm, it should be expected that ISC would provide better quality solutions when executed on SO problems having many local extrema of differing quality.)

Another very recent addition to the literature on local SO algorithms on integer-ordered spaces is DSA [Lim 2012], where the traditional Kiefer-Wolfowitz (KW) stochastic-approximation iteration [Kushner and Yin 2003] is extended to discrete sets through an appropriate partitioning of the d-dimensional Euclidean space followed by an interpolation of the objective function. DSA converges when the interpolated function satisfies conditions required for the KW iteration to converge with probability one. Most important among these imposed conditions is that the objective function defined on the original discrete space is multimodular (or equivalently L-convex [Murota 2005]). DSA and R-SPLINE can be seen (respectively) as sample-average approximation [Ruszczynski and Shapiro 2003] and stochastic-approximation methods adapted to discrete problems through the strategic extension of the underlying objective function into a continuous space.

We have deliberately omitted a more detailed description and assessment of a few other relevant algorithms, most notably OptQuest (OptTek Systems, Inc.) and Coordinate Search [Hong 2005]. The omission of the latter was prompted by early experiments, where it became evident that both COMPASS and R-SPLINE are more efficient. OptQuest, on the other hand, is a heuristic algorithm that is capable of tackling local SO algorithms on integer-ordered spaces. OptQuest, however, provides neither guarantees of convergence nor rigorous finite-time solution assessment. Furthermore, detailed experiments presented in Xu et al. [2010] demonstrate that ISC is competitive with OptQuest while providing rigorous probabilistic guarantees on solution quality.

1.2. Contributions

We develop R-SPLINE, a family of algorithms tailored to obtain high-quality locally optimal solutions to SO problems on integer-ordered feasible spaces. R-SPLINE works by identifying local minima of a sequence of appropriately constructed continuous relaxations, within a retrospective framework. R-SPLINE's efficiency comes from using a local search routine on a continuous relaxation of the objective function, and an iterative retrospective framework that is constructed to preserve efficiency through the strategic use of "warm starts" and the natural use of common random numbers.

The following are four specific contributions of this article.

(i) Like COMPASS, R-SPLINE is applicable to a relatively large class of integer-ordered problems. Linear or nonlinear deterministic constraints are allowed; at any solution x, the oracle reports infeasibility or returns a point estimate of the objective function $g(x)$. The objective function $g(x)$ can be a performance measure from either a terminating or steady-state model; in particular, the observations may be dependent and subject to initial bias. A large class of simulation point estimators of $g(x)$ is considered: sampling can be in discrete or continuous time; the point estimate need not be a mean. For example, $g(x)$ could be the steady-state standard deviation of the number of customers in a queueing system. These variations are considered by R-SPLINE without changing the R-SPLINE logic, since they are contained in the user-written simulation oracle.

(ii) R-SPLINE appears to be among the first few multidimensional gradient-based continuous-search algorithms for the specific context of solving local SO problems on integer-ordered spaces. Our numerical experience suggests that R-SPLINE compares favorably to the fastest available algorithms for solving SO problems

having unimodal objective functions, or those having multiple modes of similar value. While R-SPLINE is guaranteed to converge to a local extremum, no guarantees are provided on the quality of the local extremum when many such exist.

(iii) We demonstrate the almost-sure convergence of the iterates of R-SPLINE to a true local minimum of the function g in a certain strong sense. Specifically, we show that R-SPLINE's iterates attain a true local minimum after only a finite number of iterations, with probability one. More importantly, we show that the probability of R-SPLINE returning a solution that is not a true local minimum decays exponentially in sample size, when the sample sizes are increased at a fast enough rate and the feasible solution space is finite.

(iv) Code, along with detailed instructions for execution and some simple examples, is available through the second author's website at <https://filebox.vt.edu/users/pasupath/pasupath.htm>. The code available through the website is that used in the numerical examples of Section 7.

1.3. Organization

In Section 1.4, we provide a list of notation and terminology that is used throughout the article. In Section 2, we present the problem statement, including a rigorous definition of local minima over integer lattices, and discuss the scope of the proposed algorithm. We provide a broad overview of the proposed algorithm R-SPLINE in Section 3, followed by a detailed discussion and algorithm listing of the linear interpolation and search strategies in Section 4. In Section 5, we detail the asymptotic behavior of R-SPLINE through convergence and convergence-rate results. Section 6 contains discussion of five implementation topics. This is followed by Section 7, which contains discussion and Monte Carlo results for two families of test problems. We conclude the article with a summary and some concluding remarks in Section 8.

1.4. Notation and Terminology

The following is a list of key notation and definitions adopted in this article: (i) If $x = (x_1, \ldots, x_d)$ is a $1 \times d$ vector, then the L_2 norm of x is defined by $\|x\| = (x_1^2 + x_2^2 + \cdots + x_d^2)^{1/2}$; (ii) The d-dimensional unit vector with its ith component equal to 1 is denoted by $e_i = (0, \ldots, 1, \ldots, 0)$; (iii) The sequence of random vectors $\{X_n\}$ is said to converge to a random vector X with probability one, written as $X_n \to X$ wp1, if $\Pr\{\lim_{n\to\infty} X_n = X\} = 1$; (iv) The sequence of random vectors $\{X_n\}$ is said to converge to a random vector X in distribution, written as $X_n \Rightarrow X$, if $\lim_{n\to\infty} \Pr\{X_n \leq t\} = \Pr\{X \leq t\}$ for all continuity points t of $\Pr\{X \leq t\}$; (v) For a sequence of real numbers $\{a_n\}$, we say $a_n = O(1)$ if $\{a_n\}$ is bounded, that is, $\exists c > 0$ with $|a_n| < c$ for large enough n; also, we say $a_n = O(b_n)$ where $\{a_n\}$ and $\{b_n\}$ are real-valued sequences if the sequence $a_n/b_n = O(1)$; (vi) For a sequence of events $\{A_n\}$ defined on a probability space, the event that infinitely many of A_n occur is denoted by A_n $i.o.$ and is defined as A_n $i.o.$ $= \limsup_n A_n = \bigcap_{i=1}^{\infty} \bigcup_{j=i}^{\infty} A_j$; (vii) If D is a set, then $|D|$ is the cardinality of D.

2. PROBLEM STATEMENT

R-SPLINE seeks a *local minimum* of a real-valued function $g : \mathbb{X} \mapsto \mathbb{R}$, where the feasible region $\mathbb{X}$ is a subset of $\mathbb{Z}^d$—the set of d-dimensional integer vectors. The function g can be observed only via a Monte Carlo oracle by statistically estimating $g(x)$ for any feasible x. Feasibility is assumed to be handled directly by the oracle, which either reports that a point x is infeasible (with no uncertainty) or returns a point estimator of $g(x)$. In that sense, both the function g and the feasible region $\mathbb{X}$ arise

from the simulation oracle, and "stochastic" constraints are disallowed. At every point $x \in \mathbb{X}$,

$$g(x) = \lim_{m \to \infty} \hat{g}_m(x) \text{ wp1},$$

where m denotes some measure of computational effort, and if $x \notin \mathbb{X}$, the oracle indicates infeasibility. Here, $\hat{g}_m(x)$ is often a sample mean. Sampling can be discrete, as when $\hat{g}_m(x) = m^{-1} \sum_{i=1}^{m} Y_i(x)$; or continuous, as when $\hat{g}_m(x) = m^{-1} \int_0^m Y_t(x)\, dt$. Observations can be autocorrelated and subject to initial transient, as when g is a steady-state performance measure.

2.1. Local Minima for the Integer-Ordered Context

In continuous optimization, a local minimum is a solution x that is no worse than every feasible point y in some continuous neighborhood of x. In the present context of integer-ordered optimization, an analogous definition of a local minimum is implied after an appropriate neighborhood definition.

Definition 2.1. The neighborhood $N(0)$ of the d-dimensional origin is defined as any subset of the d-dimensional integer lattice containing the origin, that is, $N(0) \subseteq \mathbb{Z}^d$ and $0 \in N(0)$. The corresponding neighborhood of any non-zero d-dimensional integer point x is then $N(x) = \{y : (y - x) \in N(0)\}$.

Definition 2.1 does not require the points in $N(0)$ to be close (in any sense) to the origin, but in practice they typically are close in the Euclidean sense. Also, the definition fixes the neighborhood structure to be the same at every point x. As examples, and for later use, we define special notation for two particular neighborhoods. Let $N^1(x)$ denote the integer points that lie within one unit of x. Let $N^2(x)$ denote the integer points that are within 2 units from x. An obvious limitation of Definition 2.1 is that it is specific to the Euclidean norm, rendering the notion of a local minimizer norm dependent. This is unlike continuous spaces where the notion of a local neighborhood is robust to the nature of the norm in consideration.

For a given neighborhood definition N and feasible region $\mathbb{X}$, a point $x^* \in \mathbb{X}$ is an *N-local minimizer* of g if the value of the function $g(\cdot)$ at x^* is no larger than at every feasible x in the neighborhood of x^*, that is, an *N-local minimizer* is a point in $\mathbb{X}$ that satisfies $g(x) \geq g(x^*),\ \forall x \in N(x^*) \cap \mathbb{X}$.

Let $\mathbb{M}^*$ and $\mathbb{M}^*(N)$ denote the sets of all global and N-local minimizers of g over $\mathbb{X}$ respectively, that is, $\mathbb{M}^* = \{x^* : g(x^*) \leq g(x) \text{ for all } x \in \mathbb{X}\}$ and $\mathbb{M}^*(N) = \{x^* : g(x^*) \leq g(x) \text{ for all } x \in N(x^*) \cap \mathbb{X}\}$. Then, towards solidifying intuition, we note the following six properties: (i) For the smallest neighborhood, $N^0(x) = \{x\}$, every point $x \in \mathbb{X}$ is a N^0-local minimum, and so $\mathbb{M}^*(N^0) = \mathbb{X}$; (ii) The relationship between neighborhoods and the corresponding sets of local minima is monotonic: if $N_1(0) \subset N_2(0)$, then $\mathbb{M}^*(N_2) \subseteq \mathbb{M}^*(N_1)$; (iii) For any neighborhood N large enough so that $\mathbb{X} \subseteq N(x)$ for every $x \in \mathbb{X}$, every N-local minimizer is also a global minimizer; (iv) The number of N-local minimizers cannot be less than the number of global minimizers; (v) If the feasible region $\mathbb{X}$ is finite, then there is at least one global minimizer, therefore, there is at least one N-local minimizer; (vi) If the feasible region $\mathbb{X}$ is unbounded, then possibly there is no N-local minimizer.

2.2. Scope of R-SPLINE

For any objective function g defined on a subset $\mathbb{X}$ of the integer lattice $\mathbb{Z}^d$, R-SPLINE is designed to return an N^1-local minimum, that is, a solution that is no worse than the $2d$ neighboring points that are exactly one unit away. Our motivation, as noted

in the introduction, is the abundance of SO problems requiring such locally optimal solutions.

R-SPLINE provably converges (wp1) to an N^1-local minimum, with convergence happening exponentially fast when the set $\mathbb{X}$ is finite. Along with theoretical guarantees on asymptotic performance, R-SPLINE exhibits fast finite-time performance with default parameter settings.

Since R-SPLINE is designed to solve for an N^1-local minimum of the function g, N^1 is the default neighborhood definition within R-SPLINE. (All of the exposition to follow uses the N^1 neighborhood.) This choice, however, does not preclude R-SPLINE from using more-general local-neighborhood structures. In fact, all of the convergence and convergence rate results in Section 5 hold almost identically for more-general neighborhood definitions.

Toward further elaborating this issue of neighborhood choice within R-SPLINE, we mention that there is an inherent tension in algorithm design between using a large neighborhood, such as N^2, to obtain a better-quality local solution and using a small neighborhood, such as N^1, to speed the search. R-SPLINE uses the small neighborhood N^1 but couples its search with a continuous local search that has two purposes. First, the continuous search allows R-SPLINE to follow a ridge that is not parallel to an axis, something that an N^1 local search finds difficult. Second, the continuous search allows large moves, something that an N^1 local search cannot do.

3. R-SPLINE OVERVIEW

Our objective in this section is to provide a broad outline of the algorithm R-SPLINE. The quick overview is that at each retrospective iteration R-SPLINE calls SPLINE once. SPLINE repeatedly calls SPLI followed by NE. SPLI, in turn, calls the piecewise-linear interpolation, PLI, a continuous relaxation of the discrete grid.

We begin by discussing the retrospective iterations in Section 3.1. Then, in Section 3.2, we discuss SPLINE, the routine used to solve each retrospective problem, and NE, the easy-to-explain neighborhood enumeration. SPLI, which is more complicated, is discussed in Section 4.

3.1. R-SPLINE

The iterative logic of R-SPLINE, with no stopping rule, is stated in Algorithm 1.

ALGORITHM 1: R-SPLINE

Require: initial solution $x_0 \in \mathbb{X}$; sample sizes $\{m_k\}$; limits on simulation calls $\{b_k\}$
Ensure: sequence of solutions $\{X_k^*\}$

1: Set $X_0^* = x_0$.
2: **for** $k = 1, 2, \ldots$
3: X_k^* = SPLINE(X_{k-1}^*, m_k, b_k) {obtain a solution to sample-path problem P_k}
4: **end for**

R-SPLINE uses a conceptually simple framework—during the kth retrospective iteration with sample size m_k, the routine SPLINE (composed of a gradient-based search routine SPLI and a neighborhood-enumeration routine NE) attempts to identify an N^1-local minimum of the sample-path function $\hat{g}_{m_k}(\cdot)$. R-SPLINE, via SPLINE, considers the following sample-path problem P_k during the kth iteration.

$$(P_k): \quad \text{Find } X_k^* \in \mathbb{X} \text{ such that } \hat{g}_{m_k}(x) \geq \hat{g}_{m_k}(X_k^*) \quad \forall x \in N^1(X_k^*). \tag{1}$$

As can be seen from (1), the problem P_k involves generating a sample-path function using sample size m_k. The problem generation is implicit in that the sample-path function is evaluated by the simulation oracle using common random numbers at all solutions x that are selected for simulation evaluation in the search for an N^1-local optimal solution X_k^*.

R-SPLINE is similar to other retrospective algorithms in three ways. First, each sample-path problem is independent of the others, in that different random numbers are used. Second, each sample-path problem better approximates the problem of finding a local minimizer of $g(\cdot)$ in that the sample sizes m_k increase to infinity. Third, R-SPLINE uses the solution X_{k-1}^* of the sample-path problem P_{k-1} as the starting solution (i.e., "warm start") for the subsequent sample-path problem P_k. This retrospective design is aimed at enhancing overall algorithm efficiency. The early iterations use little computation, even when the search routine SPLINE evaluates many points in $\mathbb{X}$, because the sample size m_k is small. The later iterations are usually efficient because, if the retrospective framework is working as intended, the warm-start solution X_{k-1}^* is already near an N^1-local minimum for the next sample-path problem P_k, and SPLINE needs to simulate at only a few points (albeit using a large sample size) in $\mathbb{X}$ before successful termination.

R-SPLINE differs from previous retrospective algorithms [Chen and Schmeiser 2001; Jin 1998; Pasupathy 2005], which considered continuous decision variables, in that R-SPLINE's discrete context allows it to solve (in principle) each problem P_k to local optimality. The correct convergence of R-SPLINE relies on the simple fact that, under certain conditions, the set of local minima of the sample-path function $\hat{g}_{m_k}$ converges, in a certain sense to be made rigorous, to the set of local minima of the limiting function g, as the sample size m_k tends to infinity. It may thus be expected, as we demonstrate rigorously in Section 5, that the sequence of retrospective solutions $\{X_k^*\}$ also converges to the true set of local minima $\mathbb{M}^*(N^1)$ in some precise sense.

3.2. SPLINE

The sample-path problem P_k generated within the kth iteration of R-SPLINE is solved using the search routine called SPLINE. A realization of SPLINE logic is illustrated in Figure 1. The SPLINE logic, as listed in Algorithm 2, iterates between two routines: (i) SPLI (discussed in Section 4) that suggests the next candidate solution based on a continuous piecewise-linear interpolation $\overline{g}_{m_k}(\cdot)$ of the function $\hat{g}_{m_k}(\cdot)$; and (ii) NE (discussed later in this section) that moves, through neighborhood enumeration, the search a bit closer to an N^1-local sample-path minimum or verifies that the candidate solution is such a minimum.

ALGORITHM 2: SPLINE

Require: initial solution $X_{\text{new}} \in \mathbb{X}$; sample size m_k; limit on oracle calls b_k
Ensure: updated solution $X_{\text{new}} \in \mathbb{X}$; total number of oracle calls expended n

1: Set $n = 0$ {initialize number of oracle calls}
2: **repeat**
3: $[n', X_{\text{old}}, \hat{g}_{m_k}(X_{\text{old}})] = \text{SPLI}(X_{\text{new}}, m_k, b_k)$ {heuristic continuous search}
4: **if** $\hat{g}_{m_k}(X_{\text{old}}) > \hat{g}_{m_k}(X_{\text{new}})$ **then** $X_{\text{old}} = X_{\text{new}}$ {SPLI cannot cause harm}
5: $[n'', X_{\text{new}}, \hat{g}_{m_k}(X_{\text{new}})] = \text{NE}(X_{\text{old}}, m_k)$ {neighborhood enumeration}
6: Set $n = n + n' + n''$ {update the number of oracle calls}
7: **until** $\hat{g}_{m_k}(X_{\text{new}}) = \hat{g}_{m_k}(X_{\text{old}})$ or $n > b_k$ {found a local solution or timed out}

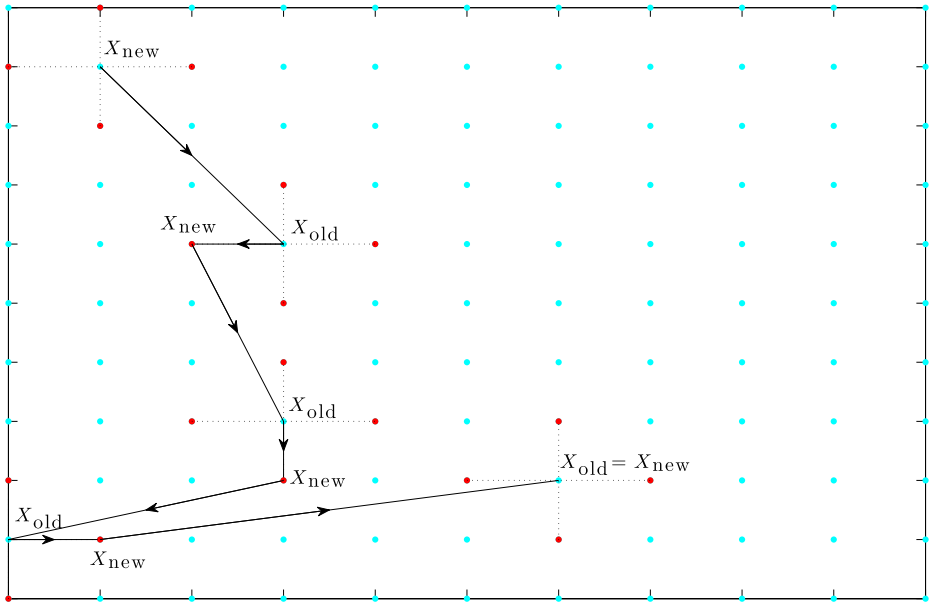

Fig. 1. A depiction of the trajectory of SPLINE, the algorithm used to solve the sample-path problems generated within R-SPLINE. SPLINE consists of a search routine SPLI and a neighborhood enumeration routine NE. SPLI is a gradient search that results in large moves (all moves from X_{new} to X_{old} in the figure) based on information provided by a piecewise-linear interpolation. NE is a local-optimality check and results in small moves (all moves X_{old} to X_{new} in the figure). SPLINE terminates if $X_{\text{old}} = X_{\text{new}}$, that is, when a local minimum for the sample-path problem has been identified.

SPLINE must be protected from an infinite loop that is sometimes caused by a "bad-realization" of the sample-path problem P_k, in contexts where the feasible region is unbounded. For protection against such cases, SPLINE counts the number of oracle calls n and terminates if an N^1-local minimum is not identified after b_k oracle calls, where the positive-valued sequence $\{b_k\} \to \infty$. As we demonstrate in Section 5, the inclusion of such limits $\{b_k\}$ on the maximum number of oracle calls during each iteration does not affect convergence.

The neighborhood-enumeration logic of routine NE is quite simple, easily discussed within this one paragraph. Every feasible solution in the N^1 neighborhood around the integer solution X_{old} is evaluated via the simulation oracle; therefore, the number of oracle calls n'' is no more than $m_k(2d+1)$. After evaluating all feasible points in the N^1-local neighborhood $N(X_{\text{old}})$, there are three cases: (i) If there is a unique best solution x, then NE returns x; (ii) If there are one or more points that are tied for best with X_{old}, then NE returns X_{old}; (iii) If there are multiple points that are tied for best and better than X_{old}, then NE returns any one of the tied solutions.

4. SEARCH WITH PIECEWISE-LINEAR INTERPOLATION (SPLI)

Recall that, in the kth retrospective iteration, R-SPLINE uses the search routine SPLINE to solve the sample-path problem P_k. SPLINE is composed of a piecewise-linear interpolation routine SPLI and a neighborhood-enumeration routine NE, which is described at the end of Section 3.2. Here, we provide more information about SPLI. Section 4.1 contains the logic of PLI, the piecewise-linear interpolation used to continuously extend the sample-path function $\hat{g}_{m_k}(\cdot)$ within SPLI. Section 4.2 contains the logic and a discussion of SPLI.

4.1. Piecewise-Linear Interpolation

We now outline the piecewise-linear interpolation strategy of PLI that is used within SPLI to extend the sample-path function $\hat{g}_{m_k}(\cdot)$ on $\mathbb{X}$ to a function $\overline{g}_{m_k}(\cdot)$ in $\mathbb{R}^d$. Among the various continuous relaxations that could be used to construct $\overline{g}_{m_k}(\cdot)$, we choose a linear interpolation that is based on the function values of $\hat{g}_{m_k}(\cdot)$ at appropriately chosen points. Specifically, consider a point $x \in \mathbb{R}^d \setminus \mathbb{Z}^d$. Consider the 2^d integer points that are the vertices of the hypercube that contains x. From among these 2^d vertices, choose $(d+1)$ vertices $x_0, x_1, \ldots, x_d$ so that the convex hull of $x_0, x_1, \ldots, x_d$ forms a simplex that contains the point x. Then, the value of $\overline{g}_{m_k}(x)$ is chosen as an appropriate convex combination of the function values $\hat{g}_{m_k}(x_0), \hat{g}_{m_k}(x_1), \ldots, \hat{g}_{m_k}(x_d)$. While other interpolations are possible, our choice of linear interpolation was dictated primarily by computational considerations, while not sacrificing continuity properties.

Any point x is contained in more than one simplex, so we make an arbitrary selection. Obtain the first vertex x_0 by taking the integer floor of each component of x; the other d vertices are obtained by sequentially adding 1 to the component with the next largest fractional part. So, if $x = (1.8, 2.3, 3.6)$, the chosen simplex will have vertices $(1,2,3), (2,2,3), (2,2,4)$ and $(2,3,4)$. Using the same selection rule for every x is necessary to ensure that the piecewise-linear relaxation $\hat{g}_{m_k}(\cdot)$ is continuous.

The piecewise-linear interpolation logic is stated in Algorithm 3. (The weights in Step 5 come from solving the convex combination of the vertices whose convex hull contains the nonintegral point x. The nice form for the weights results from the lattice structure of the feasible region.) Although not stated explicitly, m_k oracle calls at solution x are made with each mention of the point estimator $\hat{g}_{m_k}(x)$ at a new integer solution x. Also implicit in the logic is that common random numbers are used for every call to the simulation oracle within the kth retrospective iteration. Here subscripts index vectors and superscripts index components of vectors; all notation involving g is scalar.

ALGORITHM 3: Piecewise-Linear Interpolation (PLI)

Require: point $x = (x^1, x^2, \ldots x^d) \in \mathbb{R}^d \setminus \mathbb{Z}^d$; sample size m_k
Ensure: function value $\overline{g}_{m_k}(x)$, and gradient $\hat{\gamma}$ of the function $\overline{g}_{m_k}(\cdot)$ at x

1: Set $x_0^i = \lfloor x^i \rfloor$, for $i = 1, 2, \ldots, d$ {integer floor}
2: Set $z^i = x^i - x_0^i$, for $i = 1, 2, \ldots, d$ {fractional parts}
3: Sort $z = (z^1, z^2, \ldots, z^d)$ to get $1 = z^{p(0)} > z^{p(1)} \geq \cdots \geq z^{p(d)} \geq z^{p(d+1)} = 0$.
4: Set $x_i = x_{i-1} + e_{p(i)}$, for $i = 1, 2, \ldots, d$ {identify the remaining vertices}
5: Set $w^i = z^{p(i)} - z^{p(i+1)}$, for $i = 0, 1, \ldots, d$ {calculate the weight associated with each vertex i}
6: Set $n = 0$, $t = 0$, and $\overline{g}_{m_k}(x) = 0$
7: **for** $i = 0 : d$
8: **if** x_i is feasible **then** set $n = n + 1$, $t = t + w^i$, and $\overline{g}_{m_k}(x) = \overline{g}_{m_k}(x) + w^i \times \hat{g}_{m_k}(x_i)$
9: **end for**
10: **if** $t > 0$ **then** set $\overline{g}_{m_k}(x) = \overline{g}_{m_k}(x)/t$ **else** set $\overline{g}_{m_k}(x) = \infty$
11: **if** $n < d + 1$ **return** $\overline{g}_{m_k}(x)$
12: Set $\hat{\gamma}^{p(i)} = \hat{g}_{m_k}(x_i) - \hat{g}_{m_k}(x_{i-1})$, for $i = 1, 2, \ldots, d$
13: **return** $\overline{g}_{m_k}(x)$ and $\hat{\gamma}$ with ith component $\hat{\gamma}^i$, for $i = 1, 2, \ldots, d$

In addition to being piecewise linear, the interpolation function $\overline{g}_{m_k}(\cdot)$ has two notable properties: (i) By construction, it agrees with $\hat{g}_{m_k}(\cdot)$ at every feasible integer

solution; that is, $\overline{g}_{m_k}(\cdot) = \hat{g}_{m_k}(\cdot)$ for every $x \in \mathbb{X}$; and (ii) Due to (i), the (continuously) local solutions of $\overline{g}_{m_k}(\cdot)$ encompass the N^1-optimal solutions of the sample-path function $\hat{g}_{m_k}(\cdot)$.

Based on the piecewise-linear function, Algorithm 3 calculates the gradient estimate $\hat{\gamma}$ whenever all $d+1$ vertices are feasible; otherwise, no gradient estimate is computed.

4.2. SPLI

We are now ready to discuss the routine SPLI. Recall that, as discussed in Section 3.2, SPLI is the gradient-based search within SPLINE, which solves the sample-path problem P_k generated during the kth iteration of R-SPLINE. Specifically, SPLI is the search routine that is executed with the continuous extension $\overline{g}_{m_k}(\cdot)$ and which, along with the neighborhood-enumeration routine NE, identifies an N^1-local minimum of the sample-path function $\hat{g}_{m_k}$.

The SPLI logic, as listed in Algorithm 4, returns X_{best} and $\hat{g}_{m_k}(X_{\text{best}})$, where X_{best} is smallest observed feasible $\hat{g}_{m_k}(x)$) integer solution; updating X_{best} is implicit in Steps 5 and 14 of Algorithm 4. The logic moves in the direction of the gradient estimate with increasing step sizes, as shown in Step 11. As with updating X_{best}, oracle calls and the logic for common random numbers are implicit.

SPLI performs a sequence of line searches, each beginning with a gradient estimate. Step 4 adds small perturbations (uniformly distributed within a small cube centered at x) to the input integer point x_0; the routine PERTURB randomly assigns the integer solution x_0 into a touching simplex and generates an input $x_1 \in \mathbb{R}^d \setminus \mathbb{Z}^d$ to the routine PLI (in Step 5), which requires simulation at $d+1$ integer solutions, each with sample size m_k. The line search (in Steps 10–15) considers only the integer solution closest to the continuous solution, thereby avoiding simulating at $d+1$ solutions for each continuous solution. The SPLI line searches continue until a search struggles in any of four ways: (i) the estimated gradient is not defined or has zero length; (ii) the number of oracle calls exceeds the finite limit b_k; (iii) the line search requests an oracle call at an infeasible solution; or (iv) the line search moves only a short distance.

ALGORITHM 4: Search with Piecewise-Linear Interpolation (SPLI)

Require: solution $x_0 \in \mathbb{X}$; sample size m_k; call limit b_k
Ensure: solution $x \in \mathbb{X}$ with $\hat{g}_{m_k}(x) < \hat{g}_{m_k}(x_0)$, or $x = x_0$
1: **Define parameters**: initial step $s_0 = 2.0$ and multiplier $c = 2.0$
2: Set $X_{\text{best}} = x_0$ and $n' = 0$
3: Continue {begin new line search with new gradient estimate}
4: Set $x_1 = \text{PERTURB}(X_{\text{best}})$ {perturb initial solution to a non-integer point}
5: Call PLI(x_1, m_k) to observe $\overline{g}_{m_k}(x_1)$ and (possibly) gradient $\hat{\gamma} = \nabla\overline{g}_{m_k}(x_1)$
6: If $\|\hat{\gamma}\|$ is undefined or zero, **return** $[\, X_{\text{best}}, \hat{g}_{m_k}(X_{\text{best}}) \,]$ {stop if no direction}
7: Set $n' = n' + (d+1)m_k$
8: **if** $n' > b_k$ **return** $[\, X_{\text{best}}, \hat{g}_{m_k}(X_{\text{best}}) \,]$ {stop if too many oracle calls}
9: Set $i = 0$ and set $x_0 = X_{\text{best}}$
10: **repeat** {continue line search}
11: Set $i = i + 1$, set $s = c^{i-1} \times s_0$, and set $x_1 = x_0 - s \times \hat{\gamma}/\|\hat{\gamma}\|$
12: Shift x_1 to its nearest integer solution.
13: **if** x_1 is infeasible **return** $[\, X_{\text{best}}, \hat{g}_{m_k}(X_{\text{best}}) \,]$ {stop if infeasible solution}
14: Simulate at x_1 to observe $\hat{g}_{m_k}(x_1)$ and set $n' = n' + m_k$
15: **until** $x_1 \neq X_{\text{best}}$ **or** $n' > b_k$
16: If $i \leq 2$ **return** $[\, X_{\text{best}}, \hat{g}_{m_k}(X_{\text{best}}) \,]$ {stop if the line search ended quickly}
17: Go to Step 3

SPLI uses two default parameters. For the line-search procedure, SPLI uses $s_0 = 2.0$ as the initial step size and multiplies the current step size by $c = 2.0$ for the next trial point. The default initial step size $s_0 = 2.0$ is chosen to ensure a point x_1 that is outside the smallest hypercube containing x_0, but not so large that x_1 is separated from x_0 by many hypercubes. The multiplier initialization is somewhat arbitrary but ensures that not too many design points are visited on any given line search.

Consider a variation of R-SPLINE, called R-SPLINE(0), that omits calls to SPLI (or equivalently, redefines SPLI to return $X_{\text{best}} = x_0$). R-SPLINE(0) is then a simple neighborhood search in the retrospective framework. Convergence is not affected, just as it would not be affected by using any heuristic finite-time logic that returns a solution X_{best} that satisfies $\hat{g}_{m_k}(X_{\text{best}}) < \hat{g}_{m_k}(x_0)$ if $X_{\text{best}} \neq x_0$. Using SPLI is important, however, because it allows fast finite-time performance (via the gradient calculation and the resulting large steps within the line search) while increasing the likelihood of finding a higher-quality N^1-local solution, as measured by the objective function $g(\cdot)$. Both of these effects are illustrated in the bus-scheduling example in Section 7.1.

5. R-SPLINE—ASYMPTOTIC BEHAVIOR

We now describe the asymptotic behavior of R-SPLINE. Specifically, we demonstrate that, barring extreme dependence and tail behavior of the estimators $\hat{g}_{m_k}(\cdot)$, and assuming sample sizes are increased at least at a specified minimum rate, R-SPLINE converges with probability one (wp1) to the set $\mathbb{M}^*(N^1)$ of N^1-local minimizers in $\mathbb{X}$. If, in addition, the number of feasible solutions are finite (as happens when $\mathbb{X}$ is fully constrained), we show that the probability of R-SPLINE returning a solution that does not belong to $\mathbb{M}^*(N^1)$ decays exponentially in sample size. All discussion of asymptotic behavior in this section is for the context of minimization; for maximization, corresponding results hold with analogous conditions.

We now state two simplifying assumptions (Assumptions 1 and 2) that will be used when establishing convergence of R-SPLINE. The first assumption ensures that the objective function is not asymptotically flat for unbounded feasible regions. This assumption, as in Hong and Nelson [2006], is intended to preclude contexts that can make the algorithm iterates "escape to infinity" when searching for a local solution. Assumption 1 also implies that $\mathbb{M}^*$ is not empty, which in turn implies that $\mathbb{M}^*(N^1)$ is not empty.

Assumption 1. For every solution $x \in \mathbb{X}$, there exists a positive number δ (dependent on x) so that the level set $S(x, \delta) = \{x' \in \mathbb{X} : g(x') \leq g(x) + \delta\}$ is finite.

Assumption 2, stated next, is a stipulation on the minimum decay rate (across retrospective iterations) of the probability of R-SPLINE returning an N^1-local optimum of the function g. Lemmas 5.1 and 5.2 provide sufficient conditions that guarantee the satisfaction of Assumption 2. (We note here that our presentation—first stating Assumption 2 and then stating sufficient conditions for its satisfaction through Lemmas 5.1 and 5.2—is aimed at preserving generality since Assumption 2 can be satisfied in many different ways.)

Assumption 2. Let $S(x, \delta)$ be as defined in Assumption 1, and denote $S_k(x) = \{x' \in \mathbb{X} : \hat{g}_{m_k}(x') \leq \hat{g}_{m_k}(x)\}$ for $k = 1, 2, \ldots$. Given $x \in \mathbb{X}$ and $y \in \mathbb{X} \backslash S(x, \delta)$, there exists a sequence $\{q_k\}$ such that $\mathrm{P}\{y \in S_k(x)\} \leq q_k$ and $\sum_{k=1}^{\infty} q_k < \infty$.

Assumption 2 ensures that the probability of the retrospective solution sets $S_k(x)$ lying outside the set $S(x, \delta)$ decreases "fast enough" with sample size m_k, where "fast enough" is in the Borel-Cantelli sense [Billingsley 1995, pp. 59–60]. Assumption 2 is

essentially a statement on the behavior of the sample-path functions $\hat{g}_{m_k}(\cdot)$, and is satisfied under relatively mild conditions.

To support this claim, we present Lemma 5.1, which provides sufficient conditions on the estimator $\hat{g}_{m_k}(\cdot)$ to guarantee the satisfaction of Assumption 2.

LEMMA 5.1. *Let the sequence of random variables* $\{\hat{g}_{m_k}(x) - g(x)\}$ *be governed by a large-deviation principle with rate function* $I_x(s)$ *[Dembo and Zeitouni 1998]. Furthermore, let the rate function* $I_x(s)$ *be such that for any* $\epsilon > 0$, $\inf_{x\in\mathbb{X}} \min(I_x(\epsilon), I_x(-\epsilon)) = \eta > 0$. *If the sequence of sample sizes* $\{m_k\}$ *is chosen to increase to infinity faster than logarithmically, that is, if* $\limsup_{k\to\infty} m_k^{-1}(\log k)^{1+\Delta} = 0$ *for some* $\Delta > 0$, *then Assumption 2 holds.*

PROOF. From the upper bound of the Gäartner-Ellis theorem [Dembo and Zeitouni 1998, p. 53], we know that for each $z \in \mathbb{X}$ and $\epsilon > 0$, and for large enough k,

$$\frac{1}{m_k} \log \Pr\{|\hat{g}_{m_k}(z) - g(z)| > \epsilon\} \leq -\min(I_z(-\epsilon), I_z(\epsilon)) \leq -\eta < 0. \tag{2}$$

Consider $y \in \mathbb{X}\backslash S(x,\delta)$ (defined in Assumption 1). Choose $0 < \epsilon < \delta/2 \leq (g(y) - g(x))/2$, and write

$$\begin{aligned}
\Pr\{y \in S_k(x)\} &= \Pr\{\hat{g}_{m_k}(y) \leq \hat{g}_{m_k}(x)\} \\
&= \Pr\{\hat{g}_{m_k}(x) \notin (g(x)-\epsilon, g(x)+\epsilon), \hat{g}_{m_k}(y) \leq \hat{g}_{m_k}(x)\} + \\
&\qquad \Pr\{\hat{g}_{m_k}(x) \in (g(x)-\epsilon, g(x)+\epsilon), \hat{g}_{m_k}(y) \leq \hat{g}_{m_k}(x)\} \\
&\leq \Pr\{\hat{g}_{m_k}(x) \notin (g(x)-\epsilon, g(x)+\epsilon)\} + \\
&\qquad \Pr\{\hat{g}_{m_k}(y) \notin (g(y)-\epsilon, g(y)+\epsilon)\}.
\end{aligned} \tag{3}$$

From (2) and (3), we have that that, for large enough k,

$$\Pr\{y \in S_k(x)\} \leq \exp\{-m_k\eta\} = p_k. \tag{4}$$

However, since $\limsup_{k\to\infty} m_k^{-1}(\log k)^{1+\Delta} = 0$ for some $\Delta > 0$, the assertion in Lemma 5.1 holds. □

The assumed existence of a rate function in Lemma 5.1 is a mild stipulation on the tail behavior of the constituent random variables $\{\hat{g}_{m_k}(x)\}$ and their dependence [Dembo and Zeitouni 1998]. The condition $\inf_{x\in\mathbb{X}} \min(I_x(\epsilon), I_x(-\epsilon)) = \eta > 0$ is a stipulation on the behavior of the rate functions $I_x(s)$ at $s = 0$. For instance, if the second derivative $I_x''(0)$ is uniformly (in $x \in \mathbb{X}$) bounded away from 0, the condition $\inf_{x\in\mathbb{X}} \min(I_x(\epsilon), I_x(-\epsilon)) = \eta > 0$ holds. To illustrate using a specific example, consider a context where $\hat{g}_{m_k}(x)$ is the sample mean of independent and identically distributed normal random variables having mean $g(x)$ and variance $\sigma^2(x)$, i.e., $\hat{g}_{m_k}(x) = m_k^{-1}\sum_{i=1}^{m_k} Y_i(x)$ where $Y_1(x), Y_2(x), \ldots$ are iid normal with mean $g(x)$ and variance $\sigma^2(x)$. From Cramér's theorem [Bucklew 1990, p. 7], we know that $I_x(s) = s^2/(2\sigma^2(x))$. Then the condition $\inf_{x\in\mathbb{X}} \min(I_x(\epsilon), I_x(-\epsilon)) = \eta > 0$ holds if $\sup_{x\in\mathbb{X}} \sigma^2(x) < \infty$. The minimum sample-size increase condition $\limsup_{k\to\infty} m_k^{-1}(\log k)^{1+\Delta} = 0$ in Lemma 5.1 is also very mild. For instance, setting $m_k = (\log k)^2$ satisfies this stipulation. Overall, Lemma 5.1 assures us that as long as the random variables $\{\hat{g}_{m_k}(x)\}$ have light tails, do not have extreme dependence, and the sample sizes are increased fast enough, Assumption 2 is satisfied.

A slightly more intuitive set of sufficient conditions for satisfying Assumption 2 is provided in Lemma 5.2, where we assume that a central limit theorem is in force on the random variables $\{\hat{g}_{m_k}(x)\}$.

LEMMA 5.2. *Suppose a central limit theorem holds on the sequence of random variables* $\{\hat{g}_{m_k}(z)\}$ *for each* $z \in \mathbb{X}$*, that is,*

$$\sqrt{m_k}\left(\frac{\hat{g}_{m_k}(z) - g(z)}{\sigma(z)}\right) \Rightarrow Z, \tag{5}$$

where $\sigma(z) > 0$ *satisfies* $\sup_{z\in\mathbb{X}} \sigma^2(z) < \infty$*. Also, assume that as* $k \to \infty$*,*

$$\sup_t |F_{z,m_k}(t) - \Phi(t)| = O\left(\frac{1}{\sqrt{m_k}}\right), \quad \forall z \in \mathbb{X}, \tag{6}$$

where $F_{z,m_k}(\cdot)$ *denotes the cumulative distribution function of the random variable* $\sqrt{m_k}\sigma(z)^{-1}(\hat{g}_{m_k}(z) - g(z))$*. If the sequence of sample sizes* $\{m_k\}$ *satisfies* $\limsup_{k\to\infty} m_k^{-1}k^{2+\beta} = 0$ *for some* $\beta > 0$*, then Assumption* 2 *holds.*

PROOF. Consider $y \in \mathbb{X}\backslash S(x,\delta)$ (defined in Assumption 1). Choose $0 < \epsilon < \delta/2 \le (g(y) - g(x))/2$, and write

$$\begin{aligned}
\Pr\{y \in S_k(x)\} &= \Pr\{\hat{g}_{m_k}(y) \le \hat{g}_{m_k}(x)\} \\
&= \Pr\{\hat{g}_{m_k}(x) \notin (g(x)-\epsilon, g(x)+\epsilon), \hat{g}_{m_k}(y) \le \hat{g}_{m_k}(x)\} + \\
&\qquad \Pr\{\hat{g}_{m_k}(x) \in (g(x)-\epsilon, g(x)+\epsilon), \hat{g}_{m_k}(y) \le \hat{g}_{m_k}(x)\} \\
&\le \Pr\{\hat{g}_{m_k}(x) - g(x) \notin (-\epsilon,\epsilon)\} + \Pr\{\hat{g}_{m_k}(y) - g(y) \notin (-\epsilon,\epsilon)\}.
\end{aligned} \tag{7}$$

From (6) and (7), we can write

$$\begin{aligned}
\Pr\{y \in S_k(x)\} &\le F_{x,m_k}\left(-\sqrt{m_k}\frac{\epsilon}{\sigma(x)}\right) + 1 - F_{x,m_k}\left(\sqrt{m_k}\frac{\epsilon}{\sigma(x)}\right) + \\
&\qquad F_{y,m_k}\left(-\sqrt{m_k}\frac{\epsilon}{\sigma(y)}\right) + 1 - F_{y,m_k}\left(\sqrt{m_k}\frac{\epsilon}{\sigma(y)}\right) \\
&= O\left(\frac{1}{\sqrt{m_k}}\right) + 2\overline{\Phi}\left(\sqrt{m_k}\frac{\epsilon}{\sigma(x)}\right) + 2\overline{\Phi}\left(\sqrt{m_k}\frac{\epsilon}{\sigma(y)}\right) = p_k.
\end{aligned} \tag{8}$$

Now use (8), the result $\overline{\Phi}(t) = O(t^{-1}\exp\{-t^2\})$ for $t \to \infty$ [Lu and Li 2009], the assumption $\sup_{z\in\mathbb{X}} \sigma^2(z) < \infty$, and the assumed sample size increase rate, to conclude that the assertion in Lemma 5.2 holds. □

Recall that in Lemma 5.1 we stipulated light-tail behavior and precluded extreme dependence across the random variables $\hat{g}_{m_k}(x), k = 1, 2, \ldots$. In Lemma 5.2, these two stipulations are implied by the assumptions in (5) and (6). The assumptions in (5) and (6) are generally mild, and widely held in practice. For instance, when the function $g(\cdot)$ is an expectation that is estimated as the sample mean of identically distributed random variables without extreme dependence [Billingsley 1995, Section 27], such a central limit theorem holds. Similarly, the assumed bound in (6) is mild, and is directly verified through the Berry-Esséen theorem [Serfling 1980, p. 33] when the estimator $\hat{g}_{m_k}(\cdot)$ is a sample mean constructed from independent and identically distributed random variables.

The minimum sample-size increase rate stipulated through Lemma 5.2 is faster than quadratic (in iteration number), and hence more stringent than that in Lemma 5.1. This is, however, not the consequence of the respective assumptions in Lemmas 5.1 and 5.2, but instead because of the loose use of the bound in (6).

Next, we demonstrate through Lemma 5.3 that, under Assumptions 1 and 2 and for large enough k, wp1 the set of points in $\mathbb{X}$ with sample-path values superior to that of

x (denoted by $S_k(x)$) is a subset of the set of points in $\mathbb{X}$ that are truly inferior to x by at most δ (denoted by $S(x,\delta)$).

LEMMA 5.3. *If Assumptions* 1 *and* 2 *hold, the sets* $S_k(x)$ *(defined in Assumption* 2*) converge almost surely into the set* $S(x,\delta)$ *(defined in Assumption* 1*) for any* $x \in \mathbb{X}$*, that is,* $\mathrm{P}\{S_k(x) \not\subseteq S(x,\delta)\ i.o.\} = 0$.

PROOF. Under Assumption 2, $\sum_{k=1}^{\infty} \mathrm{P}\{y_k \in S_k(x)\} \leq \sum_{k=1}^{\infty} p_k < \infty$, where $y_k \in \mathbb{X}\backslash S(x,\delta)$. Then, by the first Borel-Cantelli lemma [Billingsley 1995, p. 59, 60], $\mathrm{P}\{y_k \in S_k(x)\ i.o.\} = 0$, that is, $S_k(x)$ contains points in $\mathbb{X}\backslash S(x,\delta)$ only finitely many times for $k = 1,2,\ldots$. Therefore, wp1, there exists K (dependent on δ, x, and the random realization) such that for all $k \geq K$, $S_k(x) \subseteq S(x,\delta)$. □

We now use Lemma 5.3 to prove the almost-sure convergence of R-SPLINE's iterates to a local minimum of the function g.

THEOREM 5.4. *For any neighborhood* N *and any starting solution* $x_0 \in \mathbb{X}$*, if Assumption* 1 *and Assumption* 2 *hold, then R-SPLINE generates a sequence of solutions* $\{X_k^*\}$ *that converges almost-surely to* $\mathbb{M}^*(N)$ *in the sense that* $\mathrm{P}\{X_k^* \notin \mathbb{M}^*(N)\ i.o.\} = 0$.

PROOF. SPLINE returns a solution X_k^* to P_k in finite time. Therefore, R-SPLINE returns an infinite sequence of solutions $\{X_k^*\}$. We prove that this sequence converges to $\mathbb{M}^*(N)$ wp1.

Under Assumptions 1 and 2, by Lemma 5.3, there exists K (which depends on x_0, δ, and the random realization) so that for all $k \geq K$, wp1 the sets $S_k(x_0)$ are subsets of the set $S(x_0,\delta)$. Also, the sequence of solutions $\{X_k^*\}$ has to lie in $S_k(x_0)$ because R-SPLINE guarantees to return solutions no worse than the given starting solution x_0 for every P_k. This implies, since the set $S(x_0,\delta)$ is finite by Assumption 1, that the sequence of solutions $\{X_k^*\}$ generated by R-SPLINE is bounded.

Next we prove that $\{X_k^*\}$ converges to $\mathbb{M}^*(N)$ wp1. Let $\epsilon = \min_{\mathcal{D}} |g(x') - g(x)|$, where $\mathcal{D} = \{(x,x') : x', x \in S(x_0,\delta); g(x') \neq g(x)\}$. Since $S(x_0,\delta)$ is finite, $\hat{g}_{m_k}$ uniformly converges to g wp1 on the set $S(x_0,\delta)$, as $k \to \infty$. Thus, wp1, there exists K' (dependent on ϵ, δ and the random realization) so that $|\hat{g}_{m_k}(x) - g(x)| < \epsilon/2$ for all $x \in S(x_0,\delta)$ if $k \geq K'$. This implies that if $g(x') < g(x)$, then $\hat{g}_{m_k}(x') < \hat{g}_{m_k}(x)$; $\forall x, x' \in S(x_0,\delta)$, if $k \geq K'$. This in turn implies that if $\hat{g}_{m_k}(x') \geq \hat{g}_{m_k}(x)$, then $g(x') \geq g(x)$, $\forall x, x' \in S(x_0,\delta)$, if $k \geq K'$. Thus, when k is large enough, the solutions returned by R-SPLINE lie in $S(x_0,\delta)$ wp1, and satisfy $g(X_k^*) \leq g(X_{k-1}^*)$. Since $S(x_0,\delta)$ is finite, there exists $K'' \geq \max(K,K')$ such that, after a finite number of NE and SPLI calls, R-SPLINE returns $X_k^* \in \mathbb{M}^*(N)$ wp1, if $k \geq K''$. □

Theorem 5.4 asserts that for every realization of R-SPLINE, all solutions returned during retrospective iterations past a large-enough threshold will belong to the true set of N^1-local minima. Towards providing a better sense of the likelihood of a solution obtained using R-SPLINE not being a true N^1 local minimum, we now present Theorem 5.5, which relates the probability of R-SPLINE returning a false solution (a solution that is not a true N^1-local minimum) as a function of the sample size m_k in use. The sufficient conditions that we use in Theorem 5.5 are similar to that provided in Lemma 5.1, in addition to assuming that $\mathbb{X}$ is finite.

THEOREM 5.5. *Let the sequence of random variables* $\{\hat{g}_{m_k}(x) - g(x)\}$ *be governed by a large-deviation principle with rate function* $I_x(s)$ *[Dembo and Zeitouni 1998]. Also let the feasible space* $\mathbb{X}$ *be finite. Then, the following hold.*

(i) $\Pr\{M_k^*(N^1) \not\subseteq M^*(N^1)\} = O(\exp\{-\beta m_k\})$ *for some* $\beta > 0$.

(ii) Let X_k^* be the solution returned by R-SPLINE at the end of kth iteration. If the sequence of sample sizes increases to ∞ at least linearly, that is, $\limsup_{k\to\infty} m_k^{-1}k < \infty$*, then* $\Pr\{X_k^* \notin M^*(N^1)\} = O(\exp\{-\beta k\})$ *for some* $\beta > 0$.

PROOF. Consider any set $\mathbb{W}$ satisfying $\mathbb{W} \subseteq \mathbb{X}$. Since $\mathbb{X}$ is finite, so is the set $\mathbb{W}$. Let $\mathbb{S}^*(\mathbb{W})$ and $\mathbb{S}_k^*(\mathbb{W})$ denote the sets of global minima of the functions $g(\cdot)$ and $\hat{g}_{m_k}(\cdot)$ restricted to the set $\mathbb{W}$. (The sets $\mathbb{S}^*(\mathbb{W})$ and $\mathbb{S}_k^*(\mathbb{W})$ are nonempty because $\mathbb{W}$ is finite.) Denote the minimum value attained by $g(\cdot)$ over the set $\mathbb{W}$ by $v^*(\mathbb{W})$.

Choose a point $x^*(\mathbb{W})$ so that $x^*(\mathbb{W}) \in \mathbb{S}^*(\mathbb{W})$, and a constant $\epsilon(\mathbb{W})$ so that $0 < \epsilon(\mathbb{W}) < \frac{1}{2}(\min\{g(y) : y \in \mathbb{W} \setminus \mathbb{S}^*(\mathbb{W})\} - v^*(\mathbb{W}))$. We can then write

$$\begin{aligned}\Pr\{\mathbb{S}_k^*(\mathbb{W}) \not\subseteq \mathbb{S}^*(\mathbb{W})\} &\le \sum_{y\in\mathbb{W}\setminus\mathbb{S}^*(\mathbb{W})} \Pr\{\hat{g}_{m_k}(y) \le \hat{g}_{m_k}(x^*(\mathbb{W}))\} \\ &\le \sum_{y\in\mathbb{W}\setminus\mathbb{S}^*(\mathbb{W})} \Pr\{\hat{g}_{m_k}(y) \le g(y) - \epsilon(\mathbb{W})\} \\ &\quad + \Pr\{\hat{g}_{m_k}(x^*(\mathbb{W})) \ge v^*(\mathbb{W}) + \epsilon(\mathbb{W})\}. \qquad (9)\end{aligned}$$

From the upper bound of the Gäartner-Ellis theorem [Dembo and Zeitouni 1998, p. 53] and the assumption on the structure of the rate functions across $x \in \mathbb{X}$, we know that for each $z \in \mathbb{W}$ and $\epsilon > 0$, and for large enough k,

$$\frac{1}{m_k}\log\Pr\{|\hat{g}_{m_k}(z) - g(z)| > \epsilon\} \le -\min(I_z(-\epsilon), I_z(\epsilon)) \le -\eta(z,\epsilon) < 0. \qquad (10)$$

Use (9) and (10) to write, for large enough k,

$$\begin{aligned}\Pr\{\mathbb{S}_k^*(\mathbb{W}) \not\subseteq \mathbb{S}^*(\mathbb{W})\} &\le \sum_{y\in\mathbb{W}} \exp\{-m_k\eta(y,\epsilon(\mathbb{W}))\} \\ &\le |\mathbb{W}|\exp\{-m_k\eta(\epsilon(\mathbb{W}))\}, \qquad (11)\end{aligned}$$

where $\eta(\epsilon(\mathbb{W})) = \min\{\eta(y,\epsilon(\mathbb{W})) : y \in \mathbb{W}\} > 0$.

Now write, for large enough k,

$$\begin{aligned}\Pr\{\mathbb{M}_k^*(N^1) \not\subseteq \mathbb{M}^*(N^1)\} &\le \sum_{x\in\mathbb{X}\setminus\mathbb{M}^*(N^1)} \Pr\{x \in \mathbb{S}_k^*(N^1(x))\} \\ &\le \sum_{x\in\mathbb{X}\setminus\mathbb{M}_k^*(N^1)} \Pr\{\mathbb{S}_k^*(N^1(x)) \not\subseteq \mathbb{S}^*(N^1(x))\} \\ &\le \sum_{x\in\mathbb{X}\setminus\mathbb{M}_k^*(N^1)} (2d+1)\exp\{-m_k\eta(\epsilon(N^1(x))\} \\ &\le |\mathbb{X}|(2d+1)\exp\{-m_k\eta\}, \qquad (12)\end{aligned}$$

where the third inequality in (12) follows from (11) and $\eta = \min\{\eta(\epsilon(N^1(x))) : x \in \mathbb{X} \setminus \mathbb{M}^*(N^1)\} > 0$. Conclude from (12) that the Assertion (i) holds.

Assertion (ii) follows in a straightforward manner from Assertion (i) in combination with the assumed minimum increase rate in the sequence of sample sizes. □

6. R-SPLINE IMPLEMENTATION

R-SPLINE implementation decisions, particularly on parameter values, affect finite-time performance. In the following paragraphs, we briefly discuss (i) our choice of the Monte Carlo sample size increases, (ii) our use of common random numbers (CRN),

(iii) three simplifications from earlier versions, (iv) our definition of the neighborhood evaluation NE, and (v) why our warm starts don't use the average of previous sample-path solutions. This section, being about algorithm design decisions, is intended to provide broad implementation directives that are based on empirical evidence, while not sacrificing the attractive asymptotic properties of R-SPLINE. There are no user-specified parameter values.

(i) As is evident from Section 5, the stipulations on the sample-size sequence $\{m_k\}$ to guarantee convergence are very mild—$\{m_k\} \uparrow \infty$ at least at a logarithmic rate. In general, the value of the sample size m_k during the kth iteration needs to be determined only after the $(k-1)$th sample-path problem P_{k-1} has been solved. Information from the previous sample-path problems $P_1, P_2, \ldots, P_{k-1}$ can be thus used to dynamically determine the next sample size m_k. For example, if X^*_{k-1} is equal to X^*_{k-2} then $\hat{g}_{m_{k-1}}(\cdot)$ and $\hat{g}_{m_{k-2}}(\cdot)$ are probably close, in which case R-SPLINE may want to increase the sequence $\{m_k\}$ faster. Nevertheless, R-SPLINE defaults to a static 10% increase; in particular, $m_{k+1} = \lceil 1.1 m_k \rceil$. (See Pasupathy [2010] for more on sample sizes in the general retrospective framework.)

(ii) Each randomly generated sample-path problem P_k has an objective-function surface $\hat{g}_{m_k}(\cdot)$ that is defined at every feasible x by a constant number of oracle calls m_k that begin with an implicit vector of random-number seeds ξ_k. Using the same ξ_k is desirable because any positive correlation between two responses $\hat{g}_{m_k}(x_1)$ and $\hat{g}_{m_k}(x_2)$ improves the probability that comparisons between points are correct, allowing both SPLI and NE searches to perform better. The greater correlations that arise from writing an oracle to better synchronize the use of random numbers improves the finite-time performance of R-SPLINE, but are not relevant to our convergence results. What is required, and provided easily via common random numbers, is that the value of $\hat{g}_{m_k}(x)$ must be well defined for each x; in particular, the value cannot depend on *how* SPLINE reaches x.

(iii) The original code in Wang [2009] contained three search ideas that are not retained in the code used to run the examples in Section 7 for reasons of simplicity. First, the original code maintained a list of integer solutions x that have been evaluated to obtain the value of $\hat{g}_{m_k}(x)$ to avoid resimulating the oracle at a solution. Second, the original code allowed for partial simplex evaluation; the code used in the examples uses a full simplex for gradient estimation and the nearest single integer solution for the line searches. Third, because multiple simplices enclose any continuous point, the original code allowed for a new simplex orientation at each retrospective iteration; the code used in the examples uses a fixed orientation. The simpler code is described and used in this article.

(iv) As a matter of algorithm design, NE could be defined to return any better, rather than the best, solution in the neighborhood. The convergence proof would be unaffected and, particularly in high dimensions d, the computational efficiency could be improved when the neighborhood N is big. The fundamental reason for NE to be based on the neighborhood N^1 from the problem statement is guaranteed convergence to an N^1-optimal solution. Neighborhoods larger than N^1, however, make NE slow when the number of dimensions, d, is large. Therefore, in practice we use N^1 in NE and are pleased when SPLI provides solutions better than other N^1-optimal solutions.

(v) In earlier applications of the regenerative structure, all of which have been in the context of continuous decision variables, the warm start for solving sample-path problem P_k has been based on a weighted (proportional to sample size) average of earlier solutions. Here, in a context of integer decision variables, we simply use

the previous solution, X^*_{k-1}, so that there is no issue of dealing with a non-integer solution. But using X^*_{k-1} rather than something more stable such as a weighted average has an additional advantage: it increases the chance of finding a higher quality local solution.

7. NUMERICAL EXPERIMENTS

In this section, we report the performance of R-SPLINE, ISC, and ISC-AHA on two families of problems: bus scheduling and multistage flowline with finite buffer storage. To facilitate fair comparison of the two algorithms, the budget provided to phase I (global search) of ISC was made negligible. Furthermore, the effort expended by ISC in phase III (clean-up) was not included as part of ISC's assessment. All three algorithms were used with default parameters provided as part of the algorithms. In what follows, we describe and summarize the performance of R-SPLINE and ISC on each of the two problems.

7.1. Integer Bus-Scheduling Problems

Jin [1998] studies the family of continuous bus-scheduling problems, parameterized by d, τ, and λ. Within a day of τ units of time, d buses are scheduled to pick up passengers who arrive according to a Poisson process with constant rate λ. The capacity of all buses is infinite, or at least large enough to take all waiting passengers. There is a bus arrival at times 0 and τ, the beginning and end of the day. The problem is to schedule the d buses at times $x = (x^1, x^2, \ldots, x^d)$ to minimize the passengers' expected total waiting time during the day. We restrict the bus-arrival times x to be nonnegative integers, but allow arrival times before or after the day. The objective function g is defined by a simulation oracle that returns only the observed total waiting time during one day for a given integer schedule x. Then, $\hat{g}_{m_k}(x)$ is the average of the observed total waiting time for a given schedule x, and where the averaging is performed over m_k days.

Bus-scheduling problems are appealing for comparing algorithms. They provide problem instances for any nonnegative dimension d. Their optimal solutions are known analytically for any number of buses d, and are not necessarily unique. In particular, there are $(d+1)!/[a!\,(d+1-a)!]$ integer solutions that are globally optimal, where $a = \tau - (d+1) \times \lfloor \tau/(d+1) \rfloor$. If $a \neq 0$, then optimal schedules have bus intervals $\lfloor \tau/(d+1) \rfloor$ or $\lfloor \tau/(d+1) \rfloor + 1$. For example, for the 9-bus scheduling problem with days composed of $\tau = 100$ time units and with passenger arrival rate $\lambda = 10$, the unique optimal solution is $x^* = (10, 20, 30, 40, 50, 60, 70, 80, 90)$ with $g(x^*) = 0.5\tau^2\lambda/(d+1) = 5000$ hours. The corresponding optimal solutions for the 20-bus scheduling problem are numerous although easily decipherable.

Figures 2 and 3 show sample-paths of algorithm evolution when executing R-SPLINE and ISC on the 9-bus and 20-bus scheduling problems respectively. The sample-paths plot the true, rather than estimated, objective-function values at the returned solution as a function of the expended budget. For ten or fewer dimensions, ISC and ISC-AHA are identical, so Figure 2 has two sets of twenty-five realizations, whereas Figure 3 has three sets.

As can be seen from Figures 2 and 3, R-SPLINE reaches the vicinity of the optimal solution much faster than ISC and ISC-AHA. For example, in the 9-bus problem, virtually all of the twenty-five sample-paths in R-SPLINE reach a solution whose objective function value is within 50 units of optimality after $10,000$ oracle calls. The same solution quality is not reached by most of ISC's sample-paths even after $100,000$ oracle calls. The effect is even more pronounced in the 20-bus problem, where ISC takes more than ten times as many oracle calls as does ISC-AHA and, in turn,

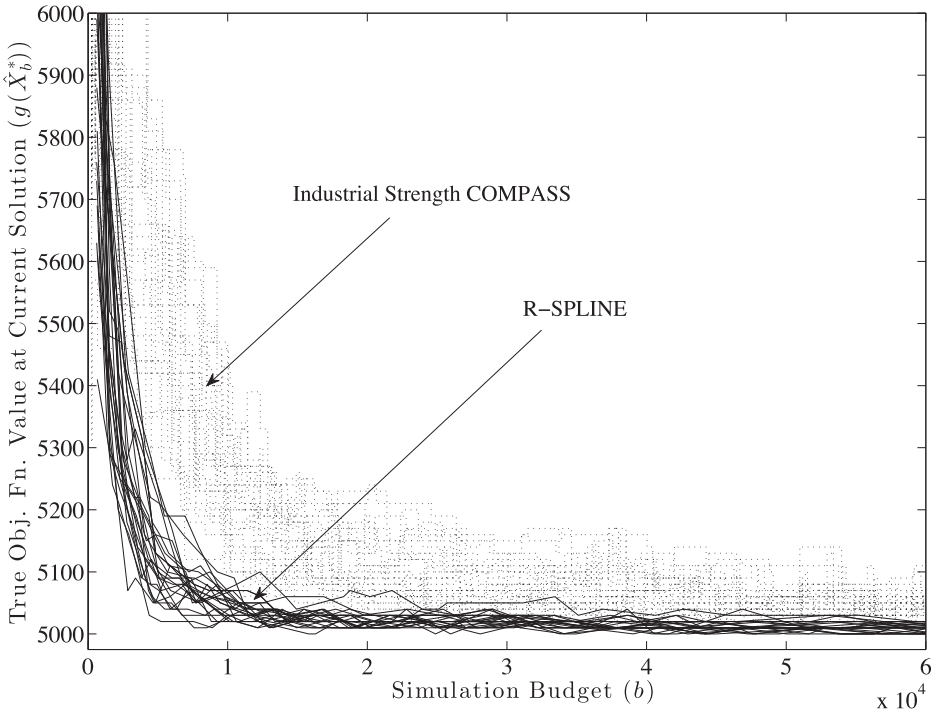

Fig. 2. Twenty-five realizations of ISC and of R-SPLINE on the 9-bus scheduling problem with $\lambda = 10$ and $\tau = 100$. The curves depict the true objective-function value at the prevailing solution, expressed as a function of the budget expended.

ISC-AHA takes about ten times as many oracle calls as does R-SPLINE to reach solutions having quality that are near optimal.

In addition to the numbers of oracle calls, it is interesting to note the relative computational overhead between the three algorithms. R-SPLINE, for instance, took about 10 seconds (on a Mac Book with a 2 GHz Intel Core 2 Duo and 4 GB memory) to solve instances of the 9-bus and 20-bus problem when allotted 100,000 oracle calls. That the difference in time taken to solve the 9-bus and 20-bus problems was negligible points to the fact that there is negligible overhead in R-SPLINE; virtually all computing effort is spent on oracle calls. The corresponding numbers for ISC were about 1.5 minutes for a 9-bus problem, and about 40 minutes for a 20-bus problem, when allotted 100,000 oracle calls. These numbers point to a heavy computational overhead in ISC, particularly as the dimension increases. This overhead is well documented [Xu et al. 2010] and is incurred due to (i) steps relating to pruning the most promising region, done by solving a series of linear programs; and (ii) generating uniformly from the most promising region, done by a sophisticated rejection algorithm [Hong et al. 2010]. ISC-AHA, which was designed to reduce the ISC overhead computations, has substantially less overhead, as documented in Xu et al. [2011] and personal communication with Jie Xu.

7.2. The Three-Stage Flowline Problem

The three-stage flowline problem appears as one of the example problems in Xu et al. [2010]. It consists of three servers in series with the second and thrid servers having

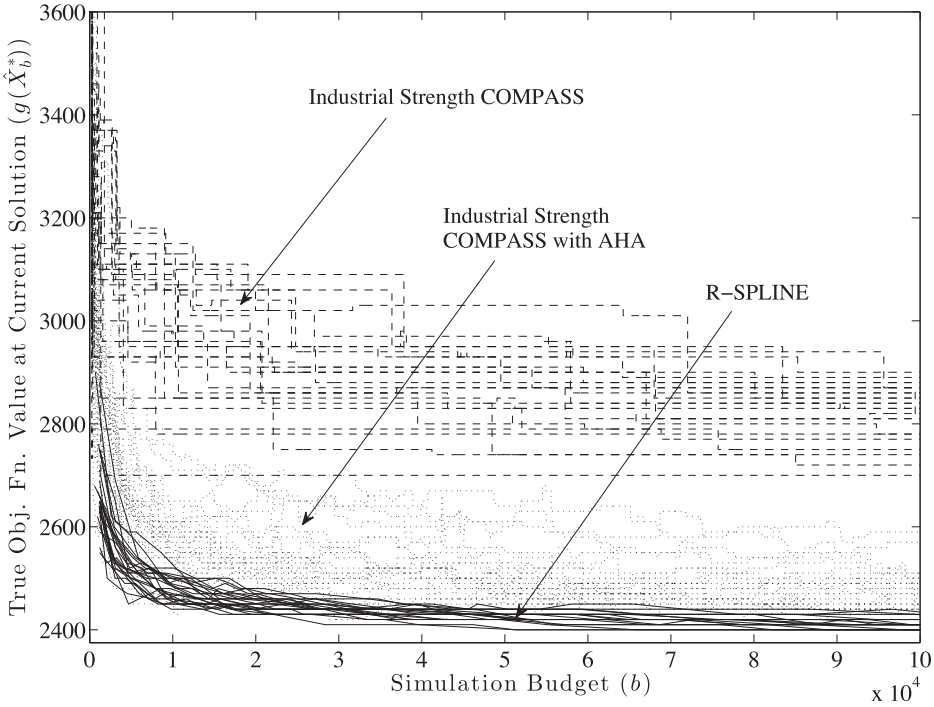

Fig. 3. Twenty-five realizations of ISC, ISC-AHA, and R-SPLINE on the 20-bus scheduling problem with $\lambda = 10$ and $\tau = 100$. The curves depict the true objective-function value at the prevailing solution, expressed as a function of the budget expended.

finite buffer capacities x_4, x_5. There are an infinite number of jobs in front of the first server. Jobs get processed sequentially through the first, second, and third servers, with exponential service times with rates $x_i, i = 1, 2, 3$, respectively. Owing to finite buffer capacities, the system suffers from communications blocking; that is, finished jobs at 1 are blocked (cannot depart) until at least one empty buffer space is available in front of server 2. Likewise, departures from server 2 are blocked until at least one empty buffer space is available in front of server 3. The total buffer space $x_4 + x_5$ and the total service rate $x_1 + x_2 + x_3$ are both restricted to 20. The objective is to identify buffer allocations $x_i, i = 4, 5$ and service rates $x_i, i = 1, 2, 3$ that maximize the expected throughput of the system, defined as the expected number of jobs leaving server 3 between the times $t = 50$ and $t = 1000$ when starting with empty servers at time $t = 0$. (This definition of expected throughput is implicit in the example considered in Xu et al. [2010] and is used as an approximation of the expected throughput at steady state.) There are two optimal solutions, $(s_1 = 6, s_2 = 7, s_3 = 7, x_1 = 12, x_2 = 8)$ and $(s_1 = 7, s_2 = 7, s_3 = 6, x_1 = 8, x_2 = 12)$, each resulting in expected throughput 5.776.

Figure 4 shows sample-paths of algorithm evolution when executing R-SPLINE and ISC on the three-stage flowline problem. As previously, the sample-paths plot the true objective-function values at the returned solution as a function of the expended budget. Consistent with numerical experience on the bus-scheduling problem, R-SPLINE's performance seems to dominate that of ISC. Both algorithms reach one of the two optimal solutions successfully but, as Figure 4 suggests, R-SPLINE reaches the vicinity of the optimal solution with about five to ten times fewer oracle calls compared to ISC.

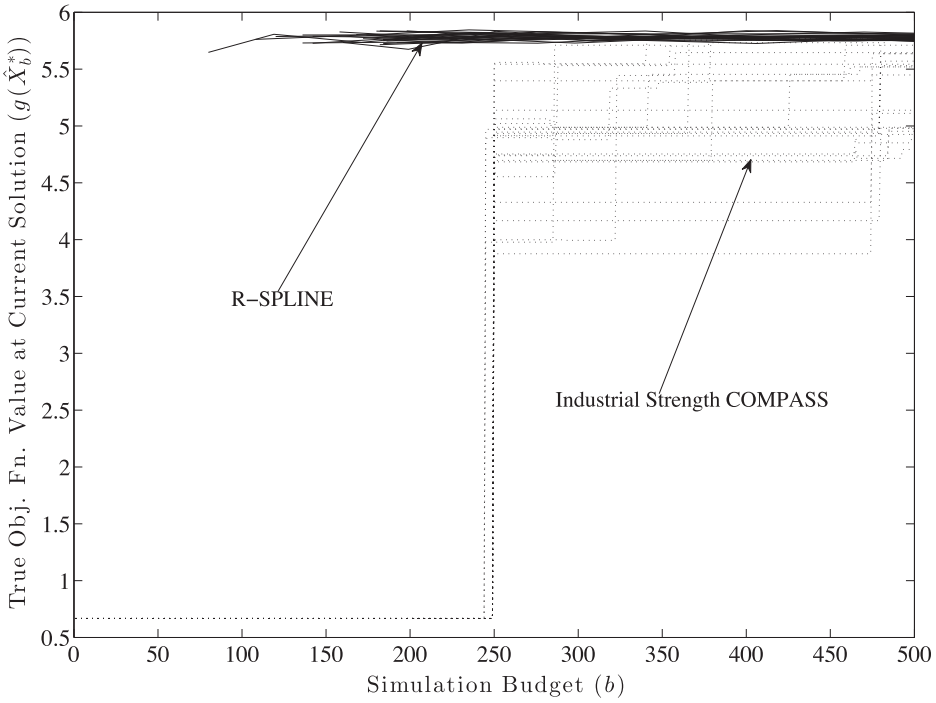

Fig. 4. Twenty-five realizations of ISC and of R-SPLINE on the three-stage flowline problem to maximize expected throughput. The curves depict the true objective-function value at the prevailing solution, expressed as a function of the budget expended.

8. SUMMARY AND CONCLUDING REMARKS

We present R-SPLINE, a retrospective algorithm for simulation-based optimization of integer-ordered stochastic systems. Sample-path problems are generated with increasing sample sizes and solved iteratively, alternating between a continuous search on a continuous relaxation (SPLI) and a discrete search (NE) on the N^1 neighborhood. The proposed continuous relaxation relies on a simplicial linear interpolation, and the continuous search is gradient based, ending whenever an estimated gradient does not directly lead to improvement. Using default algorithm parameters, R-SPLINE compares favorably to ISC and ISC-AHA, with R-SPLINE performing faster and finding a higher-quality solution more often. R-SPLINE, like ISC and ISC-AHA, requires no parameter tuning. Unlike ISC and ISC-AHA, R-SPLINE is designed for local search.

Four other specific points relating to R-SPLINE are in order.

(i) SPLINE is a search routine that is used to solve the sample-path problems generated within R-SPLINE. While SPLINE has proved to be efficient in generic examples, the proposed algorithm's framework allows the substitution of SPLINE with any other search routine that might be more appropriate for specific applications. For example, additional efficiencies can possibly be gained through the use of dynamic neighborhoods (that change size as the algorithm progresses, possibly stochastically) and the use of higher-order continuous-search algorithms. Furthermore, such additions come at little cost from the standpoint of convergence—the only stipulation for overall convergence is that the substituted search routine returns an N^1-local minimum of the sample-path problem when

the sample sizes become large enough, and spends at most a finite amount of time in every iteration.

(ii) R-SPLINE has been presented as an algorithm that (asymptotically) returns an N^1 local minimum of a function defined on an integer lattice, and observable only through a simulation oracle. While identifying N^1-local minima is our specific interest in this article, R-SPLINE seamlessly extends to more general neighborhood structures.

(iii) Our convergence analysis illustrates that, under certain conditions, the N^1-local minima sought by R-SPLINE are attained at an exponential rate with respect to the sample size in use. This is not different, however, from most discrete optimization algorithms that function in finite spaces. To this extent, the presented asymptotics still do not adequately capture the efficiencies that we have observed over competing algorithms during numerical implementation. Such efficiencies are primarily due to the incorporation of a gradient-based search—analogous to efficiencies gained when using similar techniques to identify local optima in continuous contexts.

(iv) The construction of R-SPLINE and our numerical experience suggest that R-SPLINE scales well to high dimensions. This is primarily because of the RA framework in use within R-SPLINE. Much of the searching in the design space is done in the early RA iterations when the sample size is small. The later iterations, where the sample size is larger, end with little to no searching within the design space due to the use of "warm starts" from the earlier iterations.

ACKNOWLEDGMENTS

We thank Jie Xu for pointing out the online reference for ISC-AHA, and for running the Flowline and 20-Bus problems on ISC and ISC-AHA when we had difficulty implementing the C++ codes. We thank the area editor, associate editor and the two referees for the high quality of reviews.

REFERENCES

Alrefaei, M. H. and Andradóttir, S. 2001. A modification of the stochastic ruler method for discrete stochastic optimization. *Europ. J. Oper. Res. 133*, 160–182.

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. Optimizat. 6*, 513–530.

Andradóttir, S. 2006. An overview of simulation optimization via random search. In *Simulation*, S. G. Henderson and B. L. Nelson Eds., Handbooks in Operations Research and Management Science, Elsevier, 617–631.

Atlason, J., Epelman, M. A., and Henderson, S. G. 2008. Optimizing call center staffing using simulation and analytic center cutting plane methods. *Manage. Sci. 54,* 2, 295–309.

Barton, R. R. and Meckesheimer, M. 2006. Metamodel-based simulation optimization. In *Simulation*, S. G. Henderson and B. L. Nelson Eds., Handbooks in Operations Research and Management Science, Elsevier, 535–574.

Billingsley, P. 1995. *Probability and Measure*. Wiley, New York.

Bucklew, J. A. 1990. *Large Deviation Techniques in Decision, Simulation, and Estimation*. Wiley Interscience, New York.

Chen, H. 1994. Stochastic root finding in system design. Ph.D. thesis, Purdue University, West Lafayette, IN.

Chen, H. and Schmeiser, B. W. 1994. Retrospective approximation algorithms for stochastic root finding. In *Proceedings of the Winter Simulation Conference*. J. D. Tew, S. Manivannan, D. A. Sadowski, and A. F. Seila Eds., IEEE, 255–261.

Chen, H. and Schmeiser, B. W. 2001. Stochastic root finding via retrospective approximation. *IIE Trans. 33*, 259–275.

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

Deng, G. and Ferris, M. C. 2006. Adaptation of the UOBQYA algorithm for noisy functions. In *Proceedings of the Winter Simulation Conference*. L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto Eds., IEEE, 312–319.

Deng, G. and Ferris, M. C. 2009. Variable-number sample-path optimization. *Math. Prog.* 117, 81–109.

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

Gong, W. B., Ho, Y. C., and Zhai, W. 1999. Stochastic comparison algorithm for discrete optimization with estimation. *SIAM J. Optim. 10*, 384–404.

Healy, K. and Schruben, L. W. 1991. Retrospective simulation response optimization. In *Proceedings of the Winter Simulation Conference*. B. L. Nelson, W. D. Kelton, and G. M. Clark Eds., IEEE, 901–906.

Hong, J. 2005. Discrete optimization via simulation using coordinate search. In *Proceedings of the Winter Simulation Conference*. M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines Eds., IEEE, 803–810.

Hong, J. and Nelson, B. 2006. Discrete optimization via simulation using COMPASS. *Oper. Res. 54,* 1, 115–129.

Hong, L. J. and Nelson, B. L. 2007. A framework of locally convergent random-search algorithms for discrete optimization via simulation. *ACM Trans. Model. Comput. Simul. 17*, 19/1–19/22.

Hong, L. J., Nelson, B. L., and Xu. 2010. Speeding up compass for high-dimensional discrete optimization via simulation. *Oper. Res. Lett. 38*, 550–555.

Jin, J. 1998. Retrospective optimization of stochastic systems. Ph.D. thesis, School of Industrial Engineering, Purdue University, West Lafayette, IN.

Kim, S., Pasupathy, R., and Henderson, S. G. 2012. *A Guide to SAA*. Frederick Hillier OR Series. Elsevier.

Kim, S.-H. and Nelson, B. L. 2006. Selecting the best system. In *Simulation*, S. G. Henderson and B. L. Nelson Eds., Handbooks in Operations Research and Management Science, Elsevier, 501–534.

Kushner, H. J. and Yin, G. G. 2003. *Stochastic Approximation and Recursive Algorithms and Applications*. Springer-Verlag, Berlin.

Lim, E. 2012. Stochastic approximation over multidimensional discrete sets with applications to inventory systems and admission control of queueing networks. *ACM Trans. Model. Comput. Simul. 22,* 4, 19:1–19:23.

Lu, D. and Li, W. V. 2009. A note on multivariate Gaussian estimates. *J. Math. Anal. Appl. 354*, 704–707.

Murota, K. 2005. Note on multimodularity and L-convexity. *Math. Oper. Res. 30,* 3, 658–661.

Pasupathy, R. 2010. On choosing parameters in retrospective-approximation algorithms for stochastic root finding and simulation optimization. *Oper. Res. 58*, 889–901.

Pasupathy, R. K. 2005. Retrospective-approximation algorithms for the multidimensional stochastic root-finding problem. Ph.D. thesis, Purdue University, West Lafayette, IN.

Pasupathy, R. and Henderson, S. 2006. A testbed of simulation-optimization problems. In *Proceedings of the Winter Simulation Conference*. L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto Eds., IEEE, NJ.

Pasupathy, R. and Henderson, S. G. 2011. SimOpt: A library of simulation optimization problems. In *Proceedings of the Winter Simulation Conference*. S. Jain, R. R. Creasey, J. Himmelspach, K. P. White, and M. Fu Eds., IEEE.

Pasupathy, R. and Kim, S. 2011. The stochastic root-finding problem: Overview, solutions, and open questions. *ACM Trans. Model. Comput. Simul. 21,* 3.

Pasupathy, R. and Schmeiser, B. W. 2009. Retrospective-approximation algorithms for the multidimensional stochastic root-finding problem. *ACM Trans. Model. Comput. Simul. 19,* 2, 1–36.

Pichitlamken, J. and Nelson, B. L. 2003. A combined procedure for optimization via simulation. *ACM Trans. Model. Comput. Simul. 13*, 155–179.

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

Ruszczynski, A. and Shapiro, A., Eds. 2003. *Stochastic Programming. Handbook in Operations Research and Management Science*. Elsevier, New York.

Serfling, R. J. 1980. *Approximation Theorems of Mathematical Statistics*. Wiley, New York.

Shi, L. and Ólafsson, S. 2000. Nested partitions method for stochastic optimization. *Method. Comput. Appl. Probab. 2*, 271–291.

Spall, J. C. 2003. *Introduction to Stochastic Search and Optimization*. Wiley, New York, NY.

Wang, H. 2009. Retrospective optimization of discrete stochastic systems using simplicial linear interpolation. Ph.D. thesis, School of Industrial Engineering, Purdue University, West Lafayette, IN.

Wang, H. and Schmeiser, B. W. 2008. Discrete stochastic optimization using linear interpolation. In *Proceedings of the Winter Simulation Conference*. S. J. Mason, R. R. Hill, L. Mönch, O. Rose, T. Jefferson, and J. W. Fowler Eds., IEEE, 502–508.

Whitney, J. E., Solomon, L. I., and Hill, S. D. 2001. Constrained optimization over discrete sets via SPSA with application to non-separable resource allocation. In *Proceedings of the Winter Simulation Conference*. B. A. Peters, J. S. Smith, D. J. Medeiros, and M. W. Rohrer Eds., IEEE, NJ, 313–317.

Xu, J., Hong, L. J., and Nelson, B. L. 2010. Industrial strength compass: A comprehensive algorithm and software for optimization via simulation. *ACM Trans. Model. Comput. Simul. 20*, 19/1–19/29.

Xu, J., Nelson, B. L., and Hong, L. J. 2011. An adaptive hyperbox algorithm for high-dimensional discrete optimization via simulation problems. *INFORMS J. Comput*. (Oct. 2011), iioc.1110.0481.

Yan, D. and Mukai, H. 1992. Stochastic discrete optimization. *SIAM J. Control Optimiz. 30*, 594–612.

Received August 2011; revised June 2012, December 2012; accepted January 2013