2013

On Sample Size Control in Sample Average Approximations for Solving Smooth Stochastic Programs

Royset, J.O.

On Sample Size Control in Sample Average Approximations for Solving Smooth Stochastic Programs

Johannes O. Royset
Operations Research Department, Naval Postgraduate School
Monterey, California, USA

May 29, 2012

Abstract. We consider smooth stochastic programs and develop a discrete-time optimal-control problem for adaptively selecting sample sizes in a class of algorithms based on variable sample average approximations (VSAA). The control problem aims to minimize the expected computational cost to obtain a near-optimal solution of a stochastic program and is solved approximately using dynamic programming. The optimal-control problem depends on unknown parameters such as rate of convergence, computational cost per iteration, and sampling error. Hence, we implement the approach within a receding-horizon framework where parameters are estimated and the optimal-control problem is solved repeatedly during the calculations of a VSAA algorithm. The resulting sample-size selection policy consistently produces near-optimal solutions in short computing times as compared to other plausible policies in several numerical examples.

1 Introduction

Stochastic programs that aim to minimize the expectations of random functions are rarely solvable by direct application of standard optimization algorithms. The sample average approximation (SAA) approach is a well-known framework for solving such difficult problems where a standard optimization algorithm is applied to an approximation of the stochastic program obtained by replacing the expectation by its sample average. The SAA approach is intuitive, simple, and has a strong theoretical foundation; see [2] and Chapter 5 of [46] for a summary of results, and [24, 50, 23, 1] for examples of applications. However, the framework suffers from a main difficulty: what is an appropriate sample size? A large sample size provides good accuracy in SAA, but results in a high computational cost. A small sample size is computationally inexpensive, but gives poor accuracy as the sample average only coarsely approximates the expectation. It is often difficult in practice to select a suitable sample size that balances accuracy and computational cost without extensive trial and error.

There is empirical evidence that a variable sample size during the calculations of SAA may reduce the computing time compared to a fixed sample-size policy [47, 16, 15, 3, 39, 34, 4, 30]. This is often caused by the fact that substantial objective function improvements can be achieved with small sample sizes in the early stages of the calculations. In addition, convergence of iterates to optimal and stationary solutions can typically only be ensured if the sample size is increased to infinity, see, e.g., [48]. We refer to this refinement of the SAA approach as the variable sample average approximations (VSAA) approach. There is also ample empirical evidence from other fields such as semi-infinite programming [12, 40], minimax optimization [53, 35], and optimal control [44, 6, 32] that adaptive precision-adjustment schemes may reduce computing times.

It is extremely difficult for a user to select not only one, but multiple sample sizes that overall balance computational cost and accuracy. Clearly, the number of possible sample sizes is infinite and the interaction between different stages of the calculations complicates the matter. This paper addresses the issue of how to best vary the sample size in VSAA so that a near-optimal solution can be obtain in short computing time. We develop a novel approach to sample-size selection based on discrete-time optimal control and closed-loop feedback.

While the issue of sample-size selection arises in all applications of SAA and VSAA, this paper is motivated by the specific case of smooth stochastic programs where the sample average problems are approximately solved by standard nonlinear programming algorithms. This case involves smooth sample average problems where gradients are computed relatively easily and arises for example in estimation of mixed logit models [3], search theory (see Section 5), and engineering design [38]. Important models such as two-stage stochastic programs with recourse [19], conditional Value-at-Risk minimization [36], inventory control problems [52], and complex engineering design problems [39] involve nonsmooth random functions and sample average problems. However, these nonsmooth functions can sometimes be approximated with high accuracy by smooth functions [1, 52]. Hence, the results of this paper may also be applicable in such contexts as we demonstrate in two numerical examples. Applications with integer restrictions and/or functions whose gradients may not exist or may not be easily available are beyond the scope of the paper; see [49, 42, 17, 8] for an overview of that area of research. We note that stochastic programs may also be solved by stochastic approximation [9, 21, 27] and stochastic decomposition [13, 18], which can be viewed as a version of VSAA, under suitable assumptions. In this paper we focus on VSAA without decomposition.

Existing sample-size selection policies for the VSAA approach aim at increasing the sample size sufficiently fast such that the algorithmic improvement (eventually) dominates the sampling error leading to convergence to optimal or stationary solutions [48, 47, 16, 3, 39, 30]. We also find studies of consistency of VSAA estimators defined by variable sample sizes [15].

The issue of determining a computationally efficient sample-size selection policy has received much less attention than that of asymptotic convergence. The recent paper [30] defines classes

of "optimal sample sizes" that best balance, in some asymptotic sense, sampling error and rate of convergence of the optimization algorithm used to minimize the sample average. These results provide guidance on how to choose sample sizes, but still require the user to select parameters that specify the exact sequence of sample sizes to use. We show empirically in this paper that the recommendations of [30] may be poor and highly sensitive to the selection of parameters. Consequently, we find a need for sample-size selection policies that do not require hard-to-select user specified parameters. Such policies become especially important when stochastic programs are solved as part of decision-support tools operated by personnel not trained in mathematical programming.

In [34], we eliminate essentially all user input and let a solution of an auxiliary nonlinear program determine the sample size during various stages of the calculations. The objective function of the nonlinear program is to minimize the computational cost to reach a near-optimal solution. Typically, the nonlinear program depends on unknown parameters, but computational tests indicate that even with estimates of these parameters the resulting sample-size selection policy provides reduction in computing times compared to an alternative policy. We find similar efforts to efficiently control the precision of function (and gradient) evaluations and algorithm parameters in other areas such as for instance semi-infinite programming [12], interior-point methods [20], interacting-particle algorithms [25], and simulated annealing [26].

While we here focus on obtaining a near-optimal solution, the authors of [4] deal with how to efficiently estimate the quality of a given sequence of candidate solutions. That paper provides rules for selecting variable sample sizes for that estimation at each iteration of the procedure. The rules are based on heuristically minimizing the computational cost required by the estimation procedure before a termination criterion is met. The computational effort to generate candidate solutions is not considered. The procedure requires the solution of the sample average problems to optimality, which may be computationally costly or, possibly, unattainable in finite computing time in the case of nonlinear random functions.

In this paper, we view a VSAA algorithm for solving a stochastic program as a discrete-time dynamic system subject to random disturbances due to the unknown sample averages. A similar perspective is taken in [20] in the context of interior-point methods for solving deterministic nonlinear programs and in [25] for interacting-particle algorithms. Since the VSAA approach with sample average problems solved by nonlinear programming algorithms represent a substantial departure from those contexts, we are unable to build on those studies.

We provide control inputs to the discrete-time dynamic system by selecting sample sizes for each stage of the calculations as well as the duration of each stage. Our goal is to control the system such that the expected computing time to reach a near-optimal solution of the stochastic program is minimized. As the system (i.e., the algorithm) is highly complex, we develop a surrogate model of the behavior of the system that can be used for real-time control of the system. Behavioral

models for algorithms in other areas of optimization are discussed in [29, 43]. The surrogate model leads to a surrogate discrete-time optimal-control problem in the form of a dynamic program.

While the auxiliary nonlinear program for sample-size selection in [34] is deterministic and provides no feedback about observed realizations of sample averages and algorithmic improvement, the surrogate optimal-control problem in the present paper accounts for the inherent uncertainty in VSAA and the possibility of recourse in future stages of the calculations. As the surrogate optimal-control problem depends on unknown parameters, we solve it after each stage of the calculations to utilize the latest estimates of those parameters.

We obtain the surrogate discrete-time optimal-control problem through relatively straightforward derivations, make use of approximations, and estimate several unknown parameters. In spite of this, we show in numerical examples that the sample-size selection policy generated by the optimal-control problem is consistently better than the asymptotically optimal policy of [30] and typically better than other plausible polices.

While our sample-size selection policy does depend on some user specified parameters, they are relatively easy to select and usually much easier to select than picking sequences of sample sizes directly. Hence, the proposed policy is well suited for implementation in automated decision-support tools and for use by other than experts in numerical optimization.

In section 2, we define the stochastic program considered and describe the sample-size selection problem within a VSAA algorithm as a discrete-time optimal-control problem. We show that the algorithm generates a near-optimal solution in finite time almost surely for a broad range of sample-size selections. However, the "best" sample-size selection as defined by the optimal-control problem appears difficult to determine and Section 3 defines an alternative, surrogate optimal-control problem that is tractable. The surrogate optimal-control problem depends on unknown parameters that are estimated by procedures described in Section 4. Section 4 also describes the full algorithm which integrates the surrogate optimal-control problem and the parameter estimation procedures within a receding-horizon framework. Section 5 gives a summary of numerical results.

2 Problem Statements

2.1 Stochastic Optimization Problem and Sample Average Approximations

We consider the probability space $(\Omega, \mathcal{F}, \mathbb{P})$, with $\Omega \subset \mathbb{R}^r$ and $\mathcal{F} \subset 2^\Omega$ being the Borel sigma algebra, and the *random function* $F : \mathbb{R}^d \times \Omega \to \mathbb{R}$. We let the *expected value function* $f : \mathbb{R}^d \to \mathbb{R}$ be defined by

$$f(x) := \mathbb{E}[F(x, \omega)], \tag{1}$$

where $\mathbb{E}$ denotes the expectation with respect to the known probability distribution $\mathbb{P}$. Moreover, we define the problem

$$\mathbf{P}: \quad \min_{x \in X} f(x), \tag{2}$$

where $X \subset \mathbb{R}^d$ is a convex compact set. We assume that $F(\cdot, \omega)$ is continuous on X for $\mathbb{P}$-almost every $\omega \in \Omega$ and that there exists a measurable function $C : \Omega \to \mathbb{R}$ such that $\mathbb{E}[C(\omega)] < \infty$ and $|F(x, \omega)| \leq C(\omega)$ for all $x \in X$ and $\mathbb{P}$-almost every $\omega \in \Omega$. This implies that $f(\cdot)$ is well-defined and continuous on X (see Theorem 7.43 in [46]). Hence, the optimal value of $\mathbf{P}$, denoted f^*, is defined and finite. We denote the set of optimal solutions of $\mathbf{P}$ by X^* and the set of ϵ-optimal solutions by X^*_ϵ, i.e., for any $\epsilon \geq 0$,

$$X^*_\epsilon := \{x \in X | f(x) - f^* \leq \epsilon\}. \tag{3}$$

For a general probability distribution $\mathbb{P}$, we are unable to compute $f(x)$ exactly. Hence, we approximate it using the random *sample average function* $f_N : \mathbb{R}^d \to \mathbb{R}$, $N \in \mathbb{N} := \{1, 2, 3, ...\}$, defined by

$$f_N(x) := \sum_{j=1}^{N} F(x, \omega_j)/N, \tag{4}$$

where $\omega_1, \omega_2, ..., \omega_N$ is a sample of size N consisting of independent random vectors with distribution $\mathbb{P}$. In $f_N(x)$ as well as in other expressions below, we suppress the dependence on the sample in the notation. Moreover, we denote a random vector and its realization with the same symbol. The meaning should be clear from the context.

Various sample sizes give rise to a family of (random) approximations of $\mathbf{P}$. Let $\{\mathbf{P}_N\}_{N \in \mathbb{N}}$ be this family, where, for any $N \in \mathbb{N}$, the (random) *sample average problem* $\mathbf{P}_N$ is defined by

$$\mathbf{P}_N : \quad \min_{x \in X} f_N(x). \tag{5}$$

Since $f_N(\cdot)$ is continuous on X almost surely, the minimum value of $\mathbf{P}_N$, denoted by f^*_N, is defined and finite almost surely. Let $\hat{X}^*_N$ be the set of optimal solutions of $\mathbf{P}_N$.

In this paper, we aim to approximately solve $\mathbf{P}$ by means of approximately solving a sequence of problems of the form $\mathbf{P}_N$ with varying, well-selected N. We assume that for any $N \in \mathbb{N}$ there exists a suitable algorithm for solving $\mathbf{P}_N$ given by an *algorithm map* $A_N : X \to X$ and that $A_N(\cdot)$ is a random function defined on the product space $\Omega \times \Omega \times ...$ generated by independent sampling from $\mathbb{P}$. We view $f_N(\cdot)$ as defined on the same product space. While we could state the sample-size control problem below without further assumptions, we need the following assumption about uniformly linear convergence of the algorithm map in our solution approach, where we use the abbreviation a.s. for almost surely. We find a similar linear rate of convergence assumption in [30], which also discusses other rates.

Assumption 1 *There exists a constant $\theta \in (0, 1)$ such that*

$$f_N(A_N(x)) - f^*_N \leq \theta(f_N(x) - f^*_N) \quad \text{a.s.} \tag{6}$$

for all $x \in X$ and $N \in \mathbb{N}$.

When applied to $\mathbf{P}_N$, with $F(\cdot, \omega)$ being continuously differentiable for $\mathbb{P}$-almost every $\omega \in \Omega$, gradient methods based on feasible directions typically satisfy linear rate of convergence under standard assumptions. For example, the projected gradient method with Armijo step size rule progresses at least at a linear rate in all iterations when applied to a smooth, strongly convex problem; see, e.g., Theorem 1.3.18 in [33]. Assumption 1 requires that there exists a uniform rate of convergence coefficient θ that is valid almost surely. This holds, for instance, when there exist two positive numbers $\lambda_{\min}$ and $\lambda_{\max}$ such that the eigenvalues of $f_N(x)$, for all $x \in X$ and $N \in \mathbb{N}$, belong to the interval $[\lambda_{\min}, \lambda_{\max}]$ almost surely. In the case of nonconvex problem, one cannot expect Assumption 1 to hold for all $x \in X$ but possibly only near a strict local minimum. Hence, we anticipate that the sample size recommendations derived below, which to some extent are based on Assumption 1, are most effective for convex problems and for nonconvex problems at iterates near a strict local minimum. (We examine numerically a nonconvex problem instance in Section 5 and find that the sample size recommendations also are quite effective some distance from a local minimum.)

While we in this paper focus on linearly convergent algorithm maps, the methodology is, in principle, also applicable to superlinearly convergent algorithm maps as a linear rate provides a conservative estimate of the progress of a superlinearly convergent algorithm map. However, it is beyond the scope of the paper to examine this aspect further.

It is well known that under the stated assumption on $F(\cdot, \cdot)$, independent sampling, and compactness of X, $f_N(x)$ converges to $f(x)$ uniformly on X, as $N \to \infty$, almost surely; see for example Theorem 7.48 in [46]. Now suppose that we apply an algorithm map $A_N(\cdot)$ to $\mathbf{P}_N$. Then under Assumption 1, for any $\epsilon > 0$, a sufficiently large N, and a sufficiently large number of iterations of the algorithm map, one obtains a solution in X^*_ϵ almost surely. Unfortunately, this simple approach has several drawbacks. First, if ϵ is relatively close to zero, both N and the number of iterations may be large resulting in a high computational cost. Second, since only a single sample is used, it may be difficult to estimate the variability in f^*_N and, hence, to estimate the quality of the obtained solution. Third, in practice, the algorithm map may only guarantee convergence to a global minimizer of $\mathbf{P}_N$ when starting sufficiently close to one. In such cases, the use of multiple samples "randomize" the sequence of iterates and therefore may increase the chance to obtain a good local minimum. This effect is not present when we use a single sample.

As argued above, a variable sample size may in part overcome the first drawback of the simple approach. Hence, we consider the approximate solution of a sequence of problems $\{\mathbf{P}_{N_k}\}_{k=1}^{\infty}$ with typically increasing sample sizes N_k. While we could have let the sample for $\mathbf{P}_{N_{k+1}}$ contain the sample for $\mathbf{P}_{N_k}$, we let $\mathbf{P}_{N_{k+1}}$ be independent of $\mathbf{P}_{N_k}$ for all k. This construction addresses the second and third drawbacks discussed above. Hence, we consider the following stagewise approach where at stage k an independent sample of size N_k is generated from $\mathbb{P}$. The sample of a stage is independent of the samples of previous stages. We find a similar stagewise sampling scheme in

[15]. After the sample generation, n_k iterations with the algorithm map $A_{N_k}(\cdot)$, warm started with the solution from the previous stage, are carried out on $\mathbf{P}_{N_k}$ using the generated sample. Since the iterations are warm started, n_k may often be relatively small. We view $A_{N_1}(\cdot)$, $A_{N_2}(\cdot)$, ..., and $f_{N_1}(\cdot)$, $f_{N_2}(\cdot)$, ... as random functions defined on a common probability space $\bar{\Omega}$ generated by Ω, where any element $\bar{\omega} \in \bar{\Omega}$ is of the form $\bar{\omega} = (\omega^1, \omega^2, ...)$, with $\omega^k = (\omega_1^k, \omega_2^k, ...)$, $\omega_j^k \in \Omega$, $k = 1, 2, ...,$, $j = 1, 2, ...$, being the sample for stage k. We denote the corresponding probability by $\bar{\mathbb{P}}$ and observe that this construction is possible due to the assumption about independence and the Kolmogorov consistency theorem. The approach is described in the following algorithm.

Algorithm 1 (Basic Algorithm for P)

Data. Initial solution $x_0^0 \in X$ and sample size bounds $\{(N_k^{\min}, N_k^{\max})\}_{k=1}^{\infty}$, $N_k^{\min}, N_k^{\max} \in \mathbb{N}$, $k \in \mathbb{N}$.

Step 0. Set $n_0 = 0$, $x_0^1 = x_0^0$, and stage counter $k = 1$.

Step 1a. Determine a sample size $N_k \in [N_k^{\min}, N_k^{\max}]$ and a number of iterations $n_k \geq 1$, which may depend on the previous samples $\omega^1, \omega^2, ..., \omega^{k-1}$.

Step 1b. Generate an independent sample $\{\omega_j^k\}_{j=1}^{N_k}$ from $\mathbb{P}$.

Step 2. For $i = 0$ to $n_k - 1$: Compute $x_{i+1}^k = A_{N_k}(x_i^k)$ using the sample generated in Step 1b.

Step 3. Set $x_0^{k+1} = x_{n_k}^k$, replace k by $k + 1$, and go to **Step 1a**.

The following theorem shows that Algorithm 1 generates a near-optimal solution in finite time almost surely under a relatively mild assumption on the selection of sample sizes $\{N_k\}_{k=1}^{\infty}$. The theorem requires the following assumption, which is taken from p. 393 in [46].

Assumption 2 *We assume that the following hold:*

(i) *For every $x \in X$, the moment-generating function $M_x(t) := \mathbb{E}[\exp(t(F(x, \omega) - f(x)))]$ is finite valued for all t in a neighborhood of zero.*

(ii) *There exists a measurable function $\kappa : \Omega \to [0, \infty)$ such that*

$$|F(x', \omega) - F(x, \omega)| \leq \kappa(\omega)\|x' - x\| \tag{7}$$

for all $\omega \in \Omega$ and $x', x \in X$.

(iii) *The moment-generating function $M_\kappa(t) := \mathbb{E}[\exp(t\kappa(\omega))]$ of $\kappa(\omega)$ is finite valued for all t in a neighborhood of zero.*

Theorem 1 *Suppose that Assumptions 1 and 2 hold and that the sequence* $\{x_{n_k}^k\}_{k=1}^\infty$ *is generated by Algorithm 1. If there exists a constant* $M \in \mathbb{N}$ *such that the sample size bounds* $\{(N_k^{\min}, N_k^{\max})\}_{k=1}^\infty$ *satisfy*

$$\sum_{k=1}^{\infty} \alpha^{N_k^{\min}} < \infty \tag{8}$$

for all $\alpha \in (0,1)$ *and* $N_k^{\max} \in [N_k^{\min}, N_k^{\min} + M]$ *for all* $k \in \mathbb{N}$*, then for every* $\epsilon > 0$ *there exists a* $k_\epsilon^* \in \mathbb{N}$ *such that* $x_{n_k}^k \in X_\epsilon^*$ *for all* $k \geq k_\epsilon^*$ *almost surely.*

Proof: Let $\{\tilde{N}_k^m\}_{k=1}^\infty$ be a deterministic sequence of sample sizes with $\tilde{N}_k^m = N_k^{\min} + m$, with $m \in \{0, 1, 2, ..., M\}$. First, we develop a uniform law of large numbers for $f_{\tilde{N}_k^m}(\cdot)$ as $k \to \infty$. Under Assumption 2, it follows by Theorem 7.65 in [46] that for any $\delta > 0$, there exist constants $C_m > 0$ and $\beta_m > 0$, independent of k, such that

$$\bar{\mathbb{P}}\left(\sup_{x \in X} |f_{\tilde{N}_k^m}(x) - f(x)| \geq \delta\right) \leq C_m e^{-\tilde{N}_k^m \beta_m} \tag{9}$$

for all $k \in \mathbb{N}$. Since the events $\{\sup_{x \in X} |f_{\tilde{N}_k^m}(x) - f(x)| \geq \delta\}$, $k \in \mathbb{N}$, are independent, it follows by the same arguments as in the proof of Proposition 3.1 of [15] that

$$\sum_{k=1}^{\infty} \bar{\mathbb{P}}\left(\sup_{x \in X} |f_{\tilde{N}_k^m}(x) - f(x)| \geq \delta\right) \leq \sum_{k=1}^{\infty} C_m e^{-\tilde{N}_k^m \beta_m} = C_m \sum_{k=1}^{\infty} (e^{-\beta_m})^{\tilde{N}_k^m}, \tag{10}$$

which is finite by (8). Hence, by the first Borel-Cantelli Lemma, $\bar{\mathbb{P}}(\sup_{x \in X} |f_{\tilde{N}_k^m}(x) - f(x)| \geq \delta \text{ infinitely often}) = 0$ and consequently $\sup_{x \in X} |f_{\tilde{N}_k^m}(x) - f(x)| \to 0$, as $k \to \infty$, almost surely.

Second, we examine the error at the end of the k-th stage, which we denote by $e_k := f(x_{n_k}^k) - f^*$, $k \in \mathbb{N}$. Let $\epsilon > 0$ be arbitrary and set $\gamma = (1-\theta)\epsilon/8$, where $\theta \in (0,1)$ is as in Assumption 1. Then, from above there exists a $k_\epsilon^m > 1$, $m \in \{0, 1, 2, ..., M\}$, possibly dependent on the sample $\bar{\omega} \in \overline{\Omega}$, such that $\sup_{x \in X} |f_{\tilde{N}_k^m}(x) - f(x)| \leq \gamma$ for all $k \geq k_\epsilon^m$ almost surely. Let $k_\epsilon = \max_{m=0,1,...,M} k_\epsilon^m$. Hence, when also using Assumption 1 and the fact that N_k takes on values in $[N_k^{\min}, N_k^{\max}] \subset [N_k^{\min}, N_k^{\min} + M]$, we obtain that

$$\begin{aligned}
e_k &\leq f_{N_k}(x_{n_k}^k) - f_{N_k}^* + 2\gamma \\
&\leq \theta^{n_k}[f_{N_k}(x_0^k) - f_{N_k}^*] + 2\gamma \\
&\leq \theta^{n_k}[f(x_{n_{k-1}}^{k-1}) - f^*] + 4\gamma \\
&= \theta^{n_k} e_{k-1} + 4\gamma \\
&\leq \theta e_{k-1} + 4\gamma
\end{aligned}$$

for any $k \geq k_\epsilon$ almost surely. We observe that any sequence $\{a_k\}_{k=1}^\infty$, with $a_k \in [0, \infty)$, $k \in \mathbb{N}$, constructed by the recursion $a_k = \xi a_{k-1} + b$, with $\xi \in (0,1)$ and $b \in [0, \infty)$, converges to $b/(1-\xi)$, as $k \to \infty$. Hence, there exists a $k_\epsilon^* \geq k_\epsilon$ such that $e_k \leq 8\gamma/(1-\theta)$ for all $k \geq k_\epsilon^*$ almost surely. In view of the choice of γ, the conclusion follows. □

We observe that the requirement (8) is only slightly restrictive as the minimum sample size sequences defined by $N_k^{\min} = ck$, for any $c > 0$, or by $N_k^{\min} = \sqrt{k}$ satisfy (8); see the discussion in [15]. In view of Theorem 1, many sample-size selections $\{(N_k, n_k)\}_{k=1}^{\infty}$ ensure that Algorithm 1 reaches a near-optimal solution in finite time. In this paper, however, we would like to find a selection that approximately minimizes the expected computational cost required in Algorithm 1 to reach a near-optimal solution. We refer to this problem as the sample-size control problem and formulate it as a discrete-time optimal-control problem.

We note that Algorithm 1 resembles the classical batching approach to obtain a lower bound on the optimal value of a stochastic program with recourse [24]. In that case, a number of independent sample average problems $\mathbf{P}_N$ with a fixed N are solved to optimality. In the present context, we do not assume that $F(\cdot, \omega)$ is piecewise linear or has any other structure that allows the solution of $\mathbf{P}_N$ in finite time. Moreover, we allow a variable and random sample size N_k and warm-start stages, i.e., $x_0^{k+1} = x_{n_k}^k$, in an effort to reduce the computing time to obtain a near-optimal solution.

2.2 Sample-Size Control Problem

We proceed by defining the sample-size control problem, where we need the following notation. For any sample of size $N \in \mathbb{N}$ and number of iterations n, let $A_N^n(x)$ denote the iterate after n iterations of the algorithm map $A_N(\cdot)$ initialized by x. That is, $A_N^n(x)$ is given by the recursion $A_N^0(x) = x$ and, for any $i = 0, 1, 2, ..., n-1$,

$$A_N^{i+1}(x) = A_N(A_N^i(x)). \tag{11}$$

We consider the evolution of Algorithm 1 to be a discrete-time dynamic system governed by the *dynamic equation*

$$x_{n_k}^k = A_{N_k}^{n_k}(x_{n_{k-1}}^{k-1}), k = 1, 2, 3, ..., \tag{12}$$

where $x_{n_{k-1}}^{k-1} \in X$ is the *state* at the beginning of the k-th stage, $u_k = (N_k, n_k) \in \mathbb{N} \times (\mathbb{N} \cup \{0\})$ is the *control* input for the k-th stage, and $x_{n_0}^0 = x_0^1 = x_0^0$ is the *initial condition*. The random sample of stage k, $\omega^k = (\omega_1^k, \omega_2^k, ...)$, is the *disturbance* induced at that stage. Clearly, for any $k \in \mathbb{N}$, $x_{n_k}^k$ is unknown prior to the realization of the samples ω^1, ω^2, ..., ω^k. We note that since we consider independent sampling across stages and single-point algorithm maps $A_N(\cdot)$ (i.e., maps that take as input a single point), it suffices to define the last iterate of a stage as the current state. This ensures that a new sample and the last iterate of a stage is the only required input for computing the iterates of the next stage. Multi-point algorithm maps (i.e., maps that take multiple points as input such as Quasi-Newton methods) would require an expanded state space and are not considered in this paper.

While Algorithm 1 is stated with an open-loop control of the sample size, i.e., $\{(N_k, n_k)\}_{k=1}^{\infty}$ is selected in advance, we now allow a closed-loop feedback control where the sample size and number

of iterations for a stage is determined immediately before that stage based on the observed state at the end of the previous stage. In view of the uncertainty in (12), feedback control potentially results in better selection of sample sizes and circumvents the difficulty of preselecting sample sizes. Given state $x \in X$, we define the feasible set of controls $U(x)$ as follows: If $x \in X_\epsilon^*$, then $U(x) = \{(1,0)\}$. Otherwise, $U(x) = \mathbb{N} \times \mathbb{N}$. We could also define more restrictive choices of $U(x)$ that ensure growth rules of the form (8), but do not state that in detail here. For notational convenience, we let $A_N(x) \in X_\epsilon^*$ whenever $x \in X_\epsilon^*$. That is, X_ϵ^* is a terminal state for the dynamic system (12). Let $c : \mathbb{N} \times (\mathbb{N} \cup \{0\}) \to [0, \infty)$ be the computational cost of carrying out one stage. Specifically, $c(N, n)$ is the computational cost of carrying out n iterations of algorithm map $A_N(\cdot)$, with $c(1, 0) = 0$ and $c(N, n) > 0$ for $N, n \in \mathbb{N}$.

Given an initial solution $x_0^0 \in X$, we seek a policy $\pi = \{\mu_1, \mu_2, ...\}$, where $\mu_k : X \to \mathbb{N} \times (\mathbb{N} \cup \{0\})$ with $\mu_k(x_{n_{k-1}}^{k-1}) \in U(x_{n_{k-1}}^{k-1})$ for all $x_{n_{k-1}}^{k-1} \in X$, $k \in \mathbb{N}$, that minimizes the *total cost function*

$$J_\pi(x_0^0) := \limsup_{s \to \infty} \bar{\mathbb{E}} \left[\sum_{k=1}^{s} c(\mu_k(x_{n_{k-1}}^{k-1})) \right] \tag{13}$$

subject to the constraints (12). (In (13) we slightly abuse notation by allowing $c(\cdot, \cdot)$ to take a two-dimensional vector as input instead of two scalar values.) Here, $\bar{\mathbb{E}}$ denotes expectation with respect to $\bar{\mathbb{P}}$. We assume that the cost function $c(\cdot, \cdot)$ and the policy π satisfy sufficient measurability assumptions so that this expectation is well defined.

For a given initial solution $x_0^0 \in X$, we define the *sample-size control problem*

$$\textbf{SSCP}: \quad \inf_\pi J_\pi(x_0^0), \tag{14}$$

where the infimum is over all admissible policies. Conceptually, the solution of **SSCP** provides an optimal policy that can be used in Steps 1 and 2 of Algorithm 1 to determine the next sample size and number of iterations.

Under certain assumptions including those that ensure that the terminal state X_ϵ^* is eventually reached with probability one as $N \to \infty$, the optimal value of **SSCP** is given by Bellman's equation and is computable by value iterations, and a stationary optimal policy exists; see for example Propositions 3.1.1 and 3.1.7 in [5], volume 2. However, here we focus on the practice task of generating efficient sample size policies and do not examine these issues further. There are four major difficulties with solving **SSCP**: (i) the set of ϵ-optimal solutions X_ϵ^* is typically unknown, (ii) the state space $X \subset \mathbb{R}^d$ is continuous and potentially large-dimensional, (iii) the dynamic equation (12) can only be evaluated by computationally costly calculations, and (iv) the expectation in (13) cannot generally be evaluated exactly. In the next section, we present a control scheme based on a surrogate dynamic model, receding-horizon optimization, and parameter estimation that, at least in part, overcome these difficulties.

3 Surrogate Sample-Size Control Problem

Instead of attempting to solve **SSCP**, we construct and solve a surrogate sample-size control problem in the form of a dynamic program. We base the surrogate problem on the asymptotic distributions of the progress made by the algorithm map given a particular control, which we derive next.

3.1 Asymptotic Distributions of Progress by Algorithm Map

Given a sample of size N, we consider the progress towards f_N^* after n iterations of the algorithm map $A_N(\cdot)$. It follows trivially from Assumption 1 and optimality of f_N^* that for any $x \in X$,

$$f_N^* \leq f_N(A_N^n(x)) \leq \bar{f}_N^n(x) := f_N^* + \theta^n(f_N(x) - f_N^*) \text{ a.s.} \tag{15}$$

We are unable to derive the distribution of $f_N(A_N^n(x))$, but will focus on its asymptotic distributions as well as those of its upper and lower bounds in (15). The derivations rely on the following assumptions.

Assumption 3 *We assume that* $\mathbb{E}[F(x,\omega)^2] < \infty$ *for all* $x \in X$.

Assumption 4 *There exists a measurable function* $C : \Omega \to [0,\infty)$ *such that* $\mathbb{E}[C(\omega)^2] < \infty$ *and*

$$|F(x,\omega) - F(x',\omega)| \leq C(\omega)\|x - x'\| \tag{16}$$

for all $x, x' \in X$ *and* $\mathbb{P}$*-almost every* $\omega \in \Omega$.

Below we need the following notation. Let $Y(x), x \in X$, denote normal random variables with mean zero, variance $\sigma^2(x) := Var[F(x,\omega)]$, and covariance $Cov[Y(x), Y(x')] := Cov[F(x,\omega), F(x',\omega)]$ for any $x, x' \in X$. We also let $\Rightarrow$ denote convergence in distribution.

It is well-known that the lower bound in (15) is typically "near" f^* for large N as stated next.

Proposition 1 *[45] Suppose that Assumptions 3 and 4 hold. Then,*

$$N^{1/2}(f_N^* - f^*) \Rightarrow \inf_{x \in X^*} Y(x), \tag{17}$$

as $N \to \infty$.

Consequently, if there is a unique optimal solution x^* of **P**, i.e., $X^* = \{x^*\}$, then the lower bound f_N^* on $f_N(A_N^n(x))$ (see (15)) is approximately normal with mean f^* and variance $\sigma^2(x^*)/N$ for large N.

We now turn our attention to the upper bound on $f_N(A_N^n(x))$. We present two results. The first one is an asymptotic result as $N \to \infty$ for a given n. The second one considers the situation when both N and n increase to infinity. Below we denote a normal random variable with mean m and variance v by $\mathcal{N}(m, v)$.

Theorem 2 *Suppose that Assumptions 1, 3, and 4 hold and that there is a unique optimal solution* x^* *of* $\mathbf{P}$*, i.e.,* $X^* = \{x^*\}$*. Then, for any* $x \in X$ *and* $n \in \mathbb{N}$

$$N^{1/2}[\bar{f}_N^n(x) - f^* - \theta^n(f(x) - f^*)] \Rightarrow \mathcal{N}(0, v_n(x)), \tag{18}$$

as $N \to \infty$*, where*

$$v_n(x) = (1-\theta^n)^2\sigma^2(x^*) + \theta^{2n}\sigma^2(x) + 2Cov(F(x^*,\omega), F(x,\omega))(1-\theta^n)\theta^n. \tag{19}$$

Proof: By (15),

$$\begin{aligned} N^{1/2}[\bar{f}_N^n(x) - f^* - \theta^n(f(x) - f^*)] &= (1-\theta^n)N^{1/2}(f_N^* - f^*) \\ &+ \theta^n N^{1/2}(f_N(x) - f(x)). \end{aligned} \tag{20}$$

Since $\mathbf{P}$ has a unique optimal solution, Theorem 5.7 in [46] implies that $f_N^* - f_N(x^*) = o_p(N^{-1/2})$ and, hence, $N^{1/2}(f_N^* - f_N(x^*)) \Rightarrow 0$, as $N \to \infty$. A vector-valued central limit theorem (see Theorem 29.5 in [7]) gives that $N^{1/2}(f_N(x) - f(x), f_N(x^*) - f^*) \Rightarrow (Y(x), Y(x^*))$, as $N \to \infty$. Combining these two results and the continuous mapping theorem (see Theorem 29.2 in [7]) yield

$$N^{1/2}\begin{pmatrix} f_N(x) - f(x) \\ f_N^* - f^* \end{pmatrix} \Rightarrow \begin{pmatrix} Y(x) \\ Y(x^*) \end{pmatrix}, \tag{21}$$

as $N \to \infty$. The result follows after another application of the continuous mapping theorem. □

In view of Theorem 2, we see that the upper bound on $f_N(A_N^n(x))$ is approximately normal with mean $f^* + \theta^n(f(x) - f^*)$ and variance $v_n(x)/N$ for large N. If we relax the assumption of a unique optimal solution of $\mathbf{P}$, we obtain the following asymptotic results as $n, N \to \infty$.

Theorem 3 *Suppose that Assumptions 1, 3, and 4 hold and that* $\theta^n N^{1/2} \to a \in [0, \infty]$*, as* $n, N \to \infty$*. Then, for any* $x \in X$*,*

$$\theta^{-n}[\bar{f}_N^n(x) - f^*] \Rightarrow f(x) - f^*, \text{ if } a = \infty; \tag{22}$$

$$N^{1/2}[\bar{f}_N^n(x) - f^*] \Rightarrow \inf_{x' \in X^*} Y(x') + a(f(x) - f^*), \text{ if } a \in [0, \infty); \tag{23}$$

as $N, n \to \infty$*.*

Proof: We only consider (23) as the other case follows by similar arguments. By definition,

$$\begin{aligned} & N^{1/2}[\bar{f}_N^n(x) - f^*] \\ = \; & N^{1/2}(f_N^* - f^*) + \theta^n N^{1/2}(f_N(x) - f(x)) + \theta^n N^{1/2}(f(x) - f^*) - \theta^n N^{1/2}(f_N^* - f^*). \end{aligned} \tag{24}$$

The result now follows from Proposition 1, the central limit theorem, and Slutsky's theorem (see, e.g., Exercise 25.7 of [7]). □

Corollary 1 *Suppose that Assumptions 1, 3, and 4 hold and that* $\theta^n N^{1/2} \to 0$, *as* $n, N \to \infty$. *Then, for any* $x \in X$,

$$N^{1/2}[f_N(A_N^n(x)) - f^*] \quad \Rightarrow \quad \inf_{x' \in X^*} Y(x') \tag{25}$$

as $N, n \to \infty$.

Proof: The result follows directly from (15), Proposition 1, and Theorem 3. □

In view of Theorem 3, we observe that the upper bound on $f_N(A_N^n(x))$ is approximately normally distributed with mean $f^* + \theta^n(f(x) - f^*)$ and variance $\sigma^2(x^*)/N$ for large n and N when $X^* = \{x^*\}$. Since $v_n(x) \to \sigma^2(x^*)$, as $n \to \infty$, we find that the last observation is approximately equivalent to the one after Theorem 2 when n is large. Moreover, Corollary 1 shows that the lower and upper bounds on $f_N(A_N^n(x))$, and hence also $f_N(A_N^n(x))$, have approximately the same distribution for large n and N when n is sufficiently large relative to N. In the next subsection, we adopt a conservative approach and use the upper bounds from Theorems 2 and 3 to estimate the progress of the algorithm map for different controls.

3.2 Development of Surrogate Sample-Size Control Problem

In this subsection, we model the evolution of the state $x_{n_{k-1}}^{k-1}$ using a surrogate dynamic equation based on the previous subsection and a surrogate state obtained by aggregation. We note that behavioral models of algorithmic progress exist for local search algorithms [29] and genetic algorithms [43]. However, these models do not seem to be applicable here.

Suppose that Algorithm 1 has carried out $k-1$ stages and has reached Step 1 of the k-th stage. At this point, we consider the current and future stages $l = k, k+1, k+2, ...$, in an attempt to determine the control (N_k, n_k) for the current stage. We start by considering function values instead of iterates, which aggregates the state space from d to one dimensions. Theorems 2 and 3 indicate possible models for the evolution of function values in Algorithm 1. If n_k and N_k are large, Theorem 3 states that conditional on $x_{n_{k-1}}^{k-1}$ and given a unique optimal solution of $\mathbf{P}$, an upper bound on $f_{N_k}(x_{n_k}^k)$ is approximately distributed as

$$\mathcal{N}(f^* + \theta^{n_k}(f(x_{n_{k-1}}^{k-1}) - f^*), \sigma^2(x^*)/N_k). \tag{26}$$

Moreover, if only N_k is large, Theorem 2 states that conditional on $x_{n_{k-1}}^{k-1}$, an upper bound on $f_{N_k}(x_{n_k}^k)$ is approximately distributed as

$$\mathcal{N}(f^* + \theta^{n_k}(f(x_{n_{k-1}}^{k-1}) - f^*), v_{n_k}/N_k). \tag{27}$$

We note, however, that if $\sigma(x^*) \approx \sigma(x_{n_{k-1}}^{k-1})$ and $Cov(F(x^*, \omega), F(x_{n_{k-1}}^{k-1}, \omega)) \approx \sigma(x^*)\sigma(x_{n_{k-1}}^{k-1})$, i.e., $F(x^*, \omega)$ and $F(x_{n_{k-1}}^{k-1}, \omega)$ are highly correlated, then $\sigma^2(x^*) \approx v_{n_k}$. Hence, (26) and (27) are approximately equal in distribution when $x_{n_{k-1}}^{k-1}$ is close to x^*. The paragraph after Corollary 1 indicates

that (26) and (27) are also approximately equal in distribution when n_k is large. Consequently, we adopt the simpler expression (26) as we conjecture that for small k, $x_{n_{k-1}}^{k-1}$ is far from x^* but an efficient policy typically involves a large n_k. On the other hand, when k is large, $x_{n_{k-1}}^{k-1}$ tends to be close to x^*. Hence, (26) appears to be reasonably accurate in the present context.

Ideally, we would have liked to know the distribution of $f(x_{n_k}^k)$ conditional on $f(x_{n_{k-1}}^{k-1})$, the distribution of $f(x_{n_{k+1}}^{k+1})$ conditional on $f(x_{n_k}^k)$, etc. However, such distributions appear inaccessible and we heuristically approximate them by (26), with truncation at f^* to account for the fundamental relation $f(x) \geq f^*$ for all $x \in X$. Hence, we let

$$\mathcal{N}_{\text{trunc}}(f^* + \theta^{n_k}(f(x_{n_{k-1}}^{k-1}) - f^*), \sigma^2(x^*)/N_k, f^*) \tag{28}$$

be our approximation of the distribution of $f(x_{n_k}^k)$ conditional on $f(x_{n_{k-1}}^{k-1})$, where $\mathcal{N}_{\text{trunc}}(m, v, t)$ denotes a truncated normally distributed random variable with an underlying normal distribution $\mathcal{N}(m, v)$ and lower truncation thresholds t. The cumulative distribution function of $\mathcal{N}_{\text{trunc}}(m, v, t)$ is $\Phi_{\text{trunc}}(\xi) = (\Phi((\xi - m)/\sqrt{v}) - \Phi((t - m)/\sqrt{v}))/(1 - \Phi((t - m)/\sqrt{v}))$, $\xi \geq t$, where $\Phi(\cdot)$ is the standard normal cumulative distribution function.

If $f(x_{n_{k-1}}^{k-1})$, f^*, θ, and $\sigma(x^*)$ had been known at the beginning of the k-th stage, we could use (28) to estimate $f(x_{n_k}^k)$. Moreover, we could use (28) recursively and estimate $f(x_{n_l}^l), l = k+1, k+2, ...$ In Section 4, we construct estimation schemes for f^*, θ, and $\sigma(x^*)$. Since $x_{n_{k-1}}^{k-1}$ is known at the beginning of the k-th stage, we can also estimate $f(x_{n_{k-1}}^{k-1})$ by a sample average. Hence, we proceed with (28) as the basis for our model of the evolution of $f(x_{n_l}^l), l = k, k+1, k+2, ...,$ in Algorithm 1. Specifically, we define $f_l, l = k, k+1, k+2, ...,$ to be the *surrogate state* at the beginning of the l-th stage, which represents our estimate of $f(x_{n_{l-1}}^{l-1})$. We let p_f, p^*, p_θ, and p_σ be the estimates of $f(x_{n_{k-1}}^{k-1})$, f^*, θ, and $\sigma(x^*)$, respectively. To facilitate computations, we consider a finite surrogate state space $\mathcal{F} = \{\xi_1, \xi_2, ..., \xi_{d_f}\}$ for some positive integer d_f. We let $\xi_1 = p^* + \epsilon$ as under the parameter estimate p^* of f^*, $p^* + \epsilon$ is a terminal surrogate state (see (3)) and there is no need to consider states with smaller values as they would be terminal surrogate states too. We discuss the selection of the other discretization points in Subsection 3.3.

Since (28) is a continuous random variable, we also discretize its support to obtain surrogate state transition probabilities. Specifically, given estimates p_f, p^*, p_θ, and p_σ as well as control input (N, n) and current surrogate state ξ_i, the next surrogate state is given by the random variable $\mathcal{N}_{\text{trunc}}(p^* + p_\theta^n(\xi_i - p^*), p_\sigma^2/N, p^*)$; see (28). With small exceptions due to end effects, we set the surrogate state transition probability to surrogate state ξ_j to be the probability that this random variable takes on a value in $(\xi_{j-1}, \xi_j]$. That is, the *surrogate state transition probability* from surrogate state ξ_i to surrogate state ξ_j, given control input (N, n), is expressed as

$$\eta(\xi_i, \xi_j, N, n) = \frac{\Phi([\xi_j - p^* - p_\theta^n(\xi_i - p^*)]N^{1/2}/p_\sigma) - \Phi([\xi_{j-1} - p^* - p_\theta^n(\xi_i - p^*)]N^{1/2}/p_\sigma)}{1 - \Phi(-p_\theta^n(\xi_i - p^*)N^{1/2}/p_\sigma)}, \tag{29}$$

if $i \in \{2, 3, ..., d_f\}$ and $j \in \{2, 3, ..., d_f - 1\}$, and when $j = 1$ as

$$\eta(\xi_i, \xi_1, N, n) = \frac{\Phi([\epsilon - p_\theta^n(\xi_i - p^*)]N^{1/2}/p_\sigma) - \Phi(-p_\theta^n(\xi_i - p^*)N^{1/2}/p_\sigma)}{1 - \Phi(-p_\theta^n(\xi_i - p^*)N^{1/2}/p_\sigma)}. \tag{30}$$

It is expressed as

$$\eta(\xi_i, \xi_{d_f}, N, n) = \frac{1 - \Phi([\xi_{d_f-1} - p^* - p_\theta^n(\xi_i - p^*)]N^{1/2}/p_\sigma)}{1 - \Phi(-p_\theta^n(\xi_i - p^*)N^{1/2}/p_\sigma)}, \tag{31}$$

if $i \in \{2, 3, ..., d_f\}$ and $j = d_f$. Finally, it is expressed as

$$\eta(\xi_1, \xi_j, N, n) = 1 \tag{32}$$

if $j = 1$ and zero for $j > 0$ as ξ_1 is a terminal surrogate state. In our implementation, if $\eta(\xi_i, \xi_j, N, n) \leq 10^{-6}$, we set that transition probability equal to zero and renormalize the above probabilities.

We define the feasible set of controls $R(\xi)$ in surrogate state $\xi \in \mathcal{F}$ as follows: If $\xi = \xi_1$, then $R(\xi) := \{(1, 0)\}$. Otherwise, $R(\xi) := D_N \times D_n$, where $D_N \subset \mathbb{N}$ and $D_n \subset \mathbb{N}$ are finite subsets of cardinality d_N and d_n, respectively, representing possible sample sizes and numbers of iterations. We discuss in Subsection 3.3 how to select these sets. Finally, while **SSCP** has an infinite horizon, we find it of little value to consider more than a moderate number of stages due to inaccuracy in parameter estimates. Hence, we consider $s + 1$ stages, where s is given, and include an end cost $c_{\text{end}}(\xi)$, which equals zero if $\xi = \xi_1$ and a large constant otherwise. We use 10^{20} in our implementation.

We set the per-stage computational cost function

$$c(N, n) = wNn + w^*N^*, \tag{33}$$

where the first term models the work to carry out n iterations of the algorithm map $A_N(\cdot)$ (see Step 2 of Algorithm 1), with $w > 0$ being a parameter that we estimate based on observed run times as described in Subsection 4.1. An alternative polynomial model of computational cost based on linear regression is used in [12]. However, we find (33) reasonable in the present situation as the time required to calculate $f_N(x)$ and $\nabla f_N(x)$ for a given x is linear in N. Hence, the effort required to apply the algorithm map once tends to be linear in N. The second term in (33) accounts for the effort to compute p_f, the estimate of $f(x_{n_{k-1}}^{k-1})$, which is needed to initialize the dynamic evolution of f_l as given by the transition probabilities (29)-(32). This estimate is simply $f_{N^*}(x_{n_{k-1}}^{k-1})$, where N^* is a fixed sample size. We model the effort to carry out this estimation by w^*N^*, where the parameter w^* is estimated based on observed computing times as described in Subsection 4.1. To explicitly indicate the dependence on the parameters w and w^*, we write the computational cost function as $c(N, n; w, w^*)$.

We now define the surrogate sample-size control problem. Given a stopping tolerance $\epsilon > 0$ and the estimates p_f, p^*, p_θ, p_σ, p_w, and p_w^* of $f(x_{n_{k-1}}^{k-1})$, f^*, θ, $\sigma^2(x^*)$, w, and w^*, respectively, at the beginning of stage k, we seek an admissible policy $\pi = \{\mu_k, \mu_{k+1}, ..., \mu_{k+s}\}$, where $\mu_l : \mathcal{F} \to \mathbb{N} \times \mathbb{N}$, $l = k, k+1, k+2, ..., k+s$, with $\mu_l(\xi) \in R(\xi)$ for all $\xi \in \mathcal{F}$ and $l = k, k+1, ..., k+s$, that minimizes the *total surrogate cost function*

$$J_{k,\pi}(p_f, p^*, p_\theta, p_\sigma, p_w, p_w^*, \epsilon) := E\left[\sum_{l=k}^{k+s} c(\mu_l(f_l); p_w, p_w^*) + c_{\text{end}}(f_{k+s+1})\right] \tag{34}$$

subject to the initial condition $f_k = p_f$ and the transition probabilities (29)-(32). Here, E denotes expectation with respect to those transition probabilities. Then, we define the *surrogate sample-size control problem*

$$\textbf{S-SSCP}_k(p_f, p^*, p_\theta, p_\sigma, p_w, p_w^*, \epsilon): \quad \min_\pi J_{k,\pi}(p_f, p^*, p_\theta, p_\sigma, p_w, p_w^*, \epsilon), \tag{35}$$

where the minimum is over all admissible policies. $\textbf{S-SSCP}_k$ is essentially a stochastic shortest path problem (see for example [5], Section 7.2, Vol. 1 and Chapter 2, Vol. 2) with a finite time horizon, where the goal is to reach the terminal surrogate state in minimum expected cost and where the choice of N and n influences the conditional probability mass function of the next surrogate state as given by (29)-(32). Using an instance of $\textbf{S-SSCP}_k$ occurring during the solution of QUAD described in Section 5, Figure 1 illustrates the trade-off between computational effort and a probability mass function that offers good odds for reaching the terminal surrogate state in the next stage or, at least, a much improved surrogate state. For parameters $p^* = 1329.6$, $p_\theta = 0.72$, $p_\sigma = 308$, and $\epsilon = 1.3$, the four subplots of Figure 1 give the probability mass function of the next surrogate state given that the current surrogate state $\xi_i = 1345.9$ for various choices of N and n; see (29)-(32). The upper left plot shows the situation for $N = 11,000$ and $n = 3$, which essentially guarantee a move to an improved surrogate state as almost all of the probability mass is below 1345.9. However, the probability of reaching the terminal surrogate state is slim—about 3.5%; see the left most bar. It is much more likely to land in a surrogate state around 1337. The situation is much improved when using $N = 11,000$ and $n = 17$; see the upper right subplot. The larger number of iterations makes it much more likely to reach the terminal surrogate state in the next stage (about 34%). Of course, improved likelihood of termination comes with the an increase in computing effort from $Nn = 33,000$ in the first subplot to $Nn = 187,000$. We obtain the more favorable probability mass function by increasing n. Would it be more beneficial to increase the sample size N instead? The bottom right subplot shows the situation for $N = 61718$ and $n = 3$, which require similar computing effort as the subplot above. While the variability in the next surrogate state is reduced somewhat, the chance to reach the terminal state is negligible. Clearly, in this instance, it is more favorable to use a relatively large n at the expense of a large N. Such trade-offs are automatically examined during the solution of $\textbf{S-SSCP}_k$. The bottom left subplot

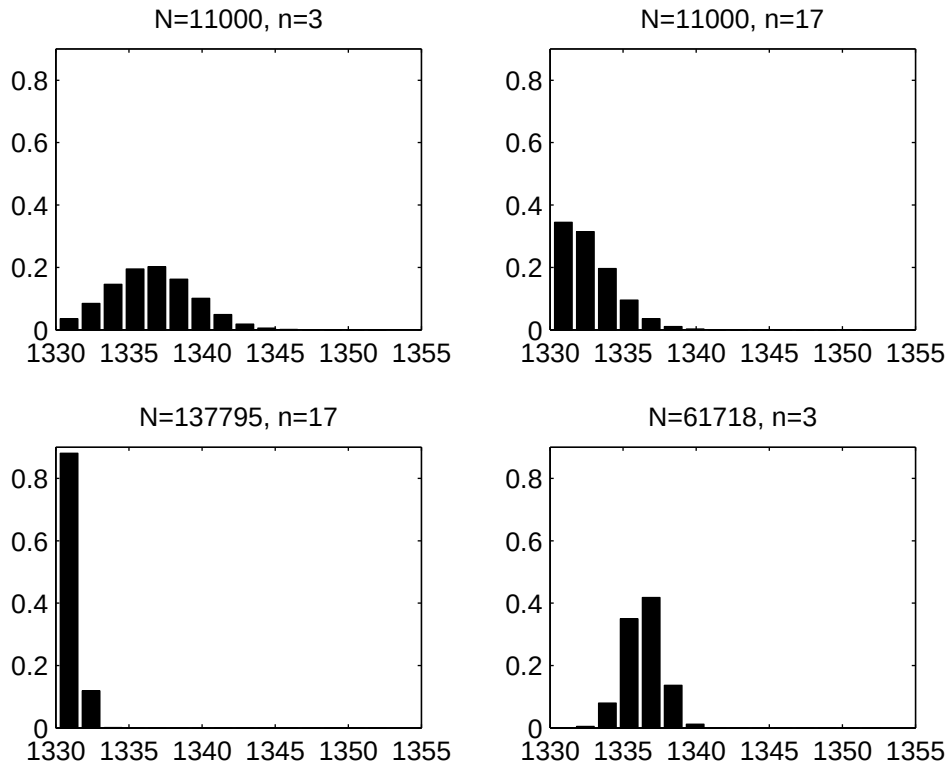

Figure 1: Example of transition probabilities in (29)-(32) from surrogate state $\xi_i = 1345.9$ for various choices of N and n under parameters $p^* = 1329.6$, $p_\theta = 0.72$, $p_\sigma = 308$, and $\epsilon = 1.3$.

shows the situation when both N and n are large, which come at a high computational cost, but almost guarantee termination in the next stage.

The next subsection discusses the solution of **S-SSCP**$_k$.

3.3 Solution of Surrogate Sample-Size Control Problem

Since the parameters p_f, p^*, p_θ, p_σ, p_w, and p_w^* may not be accurate estimates of the corresponding underlying quantities, we propose to repeatedly reestimate these parameters and resolve **S-SSCP**$_k$ as Algorithm 1 progresses. In our implementation, we opt to restimate and resolve at every stage, but other strategies are obviously also possible.

S-SSCP$_k$ is a dynamic program with d_f states, $s+1$ stages, and $d_N d_n$ possible decisions in all states except the terminal surrogate state ξ_1. Hence, the computational complexity of solving **S-SSCP**$_k$ using backward recursion is $O(sd_N d_n d_f^2)$. The solution time of **S-SSCP**$_k$ adds to the overall calculation time for Algorithm 1 and, hence, it should not be so large that it offsets the computational savings resulting from the presumably "good" selections of sample sizes given by **S-SSCP**$_k$. The threshold at which the effort to solve **S-SSCP**$_k$ outweighs its benefits is applications

dependent. In complex applications where one iteration of Algorithm 1 may take several hours, as in some engineering applications, a solution time of several minutes for **S-SSCP**$_k$ is insignificant. However, if one iteration of Algorithm 1 takes only several minutes, then **S-SSCP**$_k$ must be solved quicker. Since the solution time for **S-SSCP**$_k$ is essentially the same for complex as for simple applications, it appears that the benefits of selecting sample sizes according to **S-SSCP**$_k$ would be greater for more complex applications. However, in Section 5, we see that the benefit may also be substantial in the case of relatively simple applications.

In view of the above discussion, it is important, at least in some applications, to ensure that the solution time for **S-SSCP**$_k$ is short by selecting small integers for s, d_N, d_n, and d_f. We next discuss suitable values for $\mathcal{F}$, D_N, and D_n.

We first consider the set $\mathcal{F} = \{\xi_1, \xi_2, ..., \xi_{d_f}\}$ of discretized surrogate states. As stated above $\xi_1 = p^* + \epsilon$. Next, we include the initial state of **S-SSCP**$_k$, p_f, in $\mathcal{F}$. We see from (28) that it is unlikely to transition from p_f to a surrogate state that is much larger than p_f. Hence, we set the largest state in $\mathcal{F}$ to be $\xi_{d_f} = p_f + z_{1-\alpha_f} p_\sigma / \sqrt{N_{k-1}}$, where $z_{1-\alpha_f}$ is the $(1-\alpha_f)$-quantile of the standard normal distribution. We use $\alpha_f = 0.025$. In view of (28) with $f(x^{k-1}_{n_{k-1}})$, f^*, and $\sigma^2(x^*)$ replaced by p_f, p^*, and p^2_σ, respectively, the probability to transit from p_f to a state exceeding ξ_{d_f} is at most 0.05 regardless of the values of n_k, $N_k \geq N_{k-1}$, and $p^* \leq p_f$. Since there is a need for more accurate discretization near the terminal surrogate state ξ_1 than near the largest state ξ_{d_f}, we use $2d_f/3 + 1$ evenly spaced discretization points in the interval $[p^* + \epsilon, p_f]$ and $d_f/3$ evenly spaced discretization points for the interval $[p_f, p_f + z_{1-\alpha_f} p_\sigma / \sqrt{N_{k-1}})]$, where we ensure that d_f is divisible by 3. Certainly, other discretization schemes may also be possible including those involving segments associated with equal probabilities.

We second consider the set of possible sample sizes D_N. We include d_N integers in D_N obtained by evenly discretizing the interval $[\Delta_N^{\min} N_{k-1}, \Delta_N^{\max} N_{k-1}]$ and rounding, where we use $\Delta_N^{\min} = 1.1$ and $\Delta_N^{\max} = 100$. Hence, we allow an increase in sample size from the previous stage with as little as a factor of 1.1 or as much as a factor of 100. To reduce the possibility that the terminal surrogate state ξ_1 is not accessible for any control input, we also include in D_N a very large integer value.

We third consider the set of possible number of iterations D_n, which we obtain by evenly discretizing the interval $[3, \max\{10, \lceil \log(0.1\epsilon/(p_f - p^*))/\log p_\theta \rceil\}]$ and rounding, where $\lceil a \rceil$ denotes the smallest integer no smaller than a. We observe that the upper end of the interval is simply the larger of 10 and the number of iterations required to reach within 0.1ϵ of the optimal value in the presence of no uncertainty and the current parameter estimates.

While the above discretization of the surrogate state space spans the range of interesting surrogate states and the above restriction of possible sample sizes and numbers of iterations span the range of reasonable controls for **S-SSCP**$_k$, the resolution with which those ranges are discretized may influence the quality of the sample-size policy obtained. The number of stages s that **S-SSCP**$_k$ considers may also influence the policy obtained. We discuss these parameter choices in further

detail in Section 5.

The policy found from solving **S-SSCP**$_k$ provides controls (N_k, n_k), (N_{k+1}, n_{k+1}), (N_{k+2}, n_{k+2}), ..., (N_{k+s}, n_{k+s}). However, we utilize only (N_k, n_k) for the k-th stage as our approach is implemented within a receding-horizon framework with parameter estimation and solution of **S-SSCP**$_k$ at each stage. We refer to the resulting policy as the S-SSCP policy. We discuss the estimation of the parameters p_f, p^*, p_θ, p_σ, p_w, and p_w^* as well as the full algorithm next.

4 Parameter Estimation and Full Algorithm

In Algorithm 1, the sample-size selection $\{(N_k, n_k)\}_{k=1}^{\infty}$ is predetermined. As argued above, it is difficult to make a selection that balances computational effort with sampling accuracy and we therefore turn to **S-SSCP**$_k$ for guidance. In this section, we incorporate **S-SSCP**$_k$ into Algorithm 1 resulting in a new algorithm referred to as Algorithm 2. Algorithm 2 is essentially identical to Algorithm 1 except **S-SSCP**$_k$ determines the sample size and number of iterations of stage k. Since **S-SSCP**$_k$ relies on parameter estimates, we also include subroutines for that estimation.

We recall that Algorithm 1 consists of three main steps: 1) generate a sample of size N_k, 2) carrying out n_k iterations on a sample average problem with N_k sample points, and 3) warm start the next stage with the last iterate of the current stage. In Algorithm 2, Step 1 is expanded into two parts. First, we solve **S-SSCP**$_k$ to obtain N_k and n_k, and second we generate a sample of size N_k. Step 2 remains unchanged. Step 3 is expanded to include estimation of p_f, p^*, p_θ, p_σ, p_w, and p_w^* for the subsequent surrogate sample-size control problem **S-SSCP**$_{k+1}$, based on iterates and function values observed during stage k. The parameter estimation is carried out using six subroutines. We present these subroutines next followed by a subroutine for initializing Algorithm 2. The section ends with the complete statement of Algorithm 2.

4.1 Parameter Estimation Subroutines

After completing n_k iterations with sample size N_k in stage k of Algorithm 2, the iterates $\{x_i^k\}_{i=0}^{n_k}$ and function values $\{f_{N_k}(x_i^k)\}_{i=0}^{n_k}$ are known. We stress that these quantities are not random at that stage. Still, we retain similar notation to earlier when they were random and let the context provide the clarification. We use these quantities as well as recorded computing times of the stage to estimate the parameters p_f, p^*, p_θ, p_σ, p_w, and p_w^* for **S-SSCP**$_{k+1}$ by means of six subroutines, which we describe in turn.

The standard deviation $\sigma(x^*)$ is estimated using the following subroutine.

Subroutine A (Computes estimates p_σ of $\sigma(x^*)$)

Input. Last iterate $x_{n_k}^k$ and the sample $\{\omega_j^k\}_{j=1}^{N_k}$ of stage k.

Step 1. Compute

$$p_\sigma^2 = \frac{1}{N_k - 1} \sum_{j=1}^{N_k} (F(x_{n_k}^k, \omega_j^k) - f_{N_k}(x_{n_k}^k))^2. \tag{36}$$

Output. Standard deviation estimate p_σ.

If $x_{n_k}^k = x^*$, then p_σ^2 obviously would be the standard unbiased estimator of $\sigma(x^*)^2$. However, since this equality cannot be expected to hold, the proximity of p_σ^2 to $\sigma(x^*)^2$ cannot easily be estimated. Despite this fact, we find that p_σ^2 suffices in the present context.

We adopt the procedure in [12] to estimate the rate of convergence coefficient θ (see Assumption 1) and analyze it in detail. There is no analysis of the procedure in [12]. The procedure uses the observed function values $\{f_{N_k}(x_i^k)\}_{i=0}^{n_k}$ and an initial estimate of θ to compute an estimate of $f_{N_k}^*$. Then, a log-linear least-square regression and the estimate of $f_{N_k}^*$ generate a new estimate of θ. This process is repeated with the new estimate replacing the initial estimate of θ until the new estimate is essentially equal to the previous estimate as stated precisely next.

Subroutine B (Computes estimate $\hat{\theta}$ of rate of convergence coefficient)

Input. Previous estimate $\hat{\theta}_k$ of rate of convergence coefficient and function values $\{f_{N_k}(x_i^k)\}_{i=0}^{n_k}$ from the current stage .

Parameter. Tolerance $\epsilon_\theta > 0$.

Step 0. Set subroutine iteration counter $j = 0$ and $a_0 = \hat{\theta}_k$.

Step 1. Estimate the minimum value of $\mathbf{P}_{N_k}$ by computing

$$\phi(a_j) = \frac{1}{n_k} \sum_{i=0}^{n_k - 1} \frac{f_{N_k}(x_{n_k}^k) - a_j^{n_k - i} f_{N_k}(x_i^k)}{1 - a_j^{n_k - i}}. \tag{37}$$

Step 2. Solve the least-square problem

$$(a_{j+1}, b_{j+1}) = \arg\min_{a,b} \sum_{i=0}^{n_k} (\log(f_{N_k}(x_i^k) - \phi(a_j)) - i \log a - \log b)^2. \tag{38}$$

Step 3. If $|a_{j+1} - a_j| < \epsilon_\theta$, set $\hat{\theta} = a_{j+1}$ and **Stop**. Else, replace j by $j + 1$ and go to **Step 1**.

Output. Rate of convergence coefficient estimate $\hat{\theta}$.

The following lemma explains Step 1 of Subroutine B and deals with the same probability space as Assumption 1; see the preceding paragraph to that assumption.

Lemma 1 *Suppose that Assumption 1 holds for algorithm map $A_N(\cdot)$ with rate of convergence coefficient $\theta \in [0,1)$. If $\{x_i\}_{i=0}^n$ is generated by the recursion $x_{i+1} = A_N(x_i)$, $i = 0,1,2,...,$ with $x_0 \in X$, then,*

$$f_N^* \geq \frac{f_N(x_n) - a^{n-i} f_N(x_i)}{1 - a^{n-i}} \quad \text{a.s.} \tag{39}$$

for any $i = 0,1,...,n-1$, $a \in [\theta, 1)$, $n \in \mathbb{N}$, and $N \in \mathbb{N}$.

Proof: By Assumption 1 and the fact that $a \in [\theta, 1)$,

$$f_N(x_n) - f_N^* \leq \theta^{n-i}(f_N(x_i) - f_N^*) \leq a^{n-i}(f_N(x_i) - f_N^*) \quad \text{a.s.} \tag{40}$$

The conclusion then follows by isolating f_N^*. □

It follows from Lemma 1 that if $\{f_{N_k}(x_i^k)\}_{i=0}^{n_k}$ are generated using an algorithm map that satisfies Assumption 1 with rate of convergence coefficient θ and $a_j \geq \theta$, then Step 1 in Subroutine B averages lower bounds on $f_{N_k}^*$ to obtain an estimate, denoted by $\phi(a_j)$, of $f_{N_k}^*$.

Given $\phi(a_j)$, the estimate of $f_{N_k}^*$, Step 2 of Subroutine B computes the rate of convergence coefficient that best fits $\{f_{N_k}(x_i^k)\}_{i=0}^{n_k}$ in a least-square sense. Specifically, we use the regression model

$$e(i) := a^i b \tag{41}$$

to estimate the distance $f_{N_k}(x_i^k) - f_{N_k}^*$ after iteration i, where a and b are unknown regression coefficients estimated based on the data set $\{(i, f_{N_k}(x_i^k) - \phi(a_j))\}_{i=0}^{n_k}$. Using a logarithmic transformation, we easily obtain the values of the transformed regression coefficients $\log a$ and $\log b$ by linear least-square regression; see (38). The corresponding values of a and b are denoted by a_{j+1} and b_{j+1}.

Subroutine B is stated in [12] without any proof about its convergence. The authors' incorrectly claim that it provides the correct rate of convergence θ given that the sequence $\{f_{N_k}(x_i^k)\}_{i=0}^{n_k}$ is exactly linear, i.e., equality holds in Assumption 1. While we find that Subroutine B yields reasonable estimates of the rate of convergence in numerical examples, the situation is more complicated than stated in [12] as the below analysis shows.

We view Subroutine B as a fixed-point iteration and adopt the following notation. Given the observations $\{f_{N_k}(x_i^k)\}_{i=0}^{n_k}$, we view the calculations in Steps 1 and 2 of Subroutine B as a function $g : \mathbb{R} \to \mathbb{R}$ that takes as input an estimate a_j of the rate of convergence coefficient and returns another estimate a_{j+1}. We note that $g(\cdot)$ obviously depends on the observations $\{f_{N_k}(x_i^k)\}_{i=0}^{n_k}$ even though it is not indicated by the notation. The properties of $g(\cdot)$ explain the performance of Subroutine B as we see next. The proofs of the below results are given in the appendix due to their lengths. We first show that Steps 1 and 2 of Subroutine B are given by a relatively simple formula.

Proposition 2 *Suppose that* $\{f_{N_k}(x_i^k)\}_{i=0}^{n_k}$, *with* $n_k > 1$, *satisfies* $f_{N_k}(x_i^k) > f_{N_k}(x_{i+1}^k)$ *for all* $i = 0, 1, ..., n_k - 1$. *Then, for any* $a \in (0, 1)$,

$$g(a) = \prod_{i=n_k^0}^{n_k} \left(\frac{f_{N_k}(x_i^k) - \phi(a)}{f_{N_k}(x_{n_k-i}^k) - \phi(a)} \right)^{\alpha_i} \in (0, 1), \tag{42}$$

where $n_k^0 := \lfloor n_k/2 \rfloor + 1$, *with* $\lfloor n_k/2 \rfloor$ *being the largest integer no larger than* $n/2$, $\alpha_i := 12(i - n_k/2)/(n_k^3 + 3n_k^2 + 2n_k)$, *and* $\phi(a)$ *is as in (37) with* a_j *replaced by* a.

Proof: See Appendix. □

For notational convenience, we define $g(0) = 0$ and $g(1) = 1$. The next theorem states that Subroutine B converges to a fixed point of $g(\cdot)$, which implies that Subroutine B terminates after a finite number of iterations.

Theorem 4 *Suppose that* $\{f_{N_k}(x_i^k)\}_{i=0}^{n_k}$, *with* $n_k > 1$, *satisfies* $f_{N_k}(x_i^k) > f_{N_k}(x_{i+1}^k)$ *for all* $i = 0, 1, ..., n_k - 1$. *For any* $a_0 \in (0, 1)$, *the sequence of iterates* $\{a_j\}_{j=0}^{\infty}$ *generated by the recursion* $a_{j+1} = g(a_j)$, $j = 0, 1, 2, ...$, *converges to a fixed point* $a^* \in [0, 1]$ *of* $g(\cdot)$, *i.e.,* $a^* = g(a^*)$. *Moreover, if* $\hat{\theta}_k \in (0, 1)$, *Subroutine B terminates in finite time for any* $\epsilon_\theta > 0$.

Proof: See Appendix. □

We observe that the assumptions in Proposition 2 and Theorem 4 are rather weak. Subroutine B is guaranteed to terminate when the algorithm map generates descent in the objective function value in each iteration, which is typical for standard nonlinear programming algorithms. If the sequence of function values $\{f_{N_k}(x_i^k)\}_{i=0}^{n_k}$ is exactly linear with rate of convergence coefficient $\theta_{N_k} \in (0, 1)$, then θ_{N_k} is a fixed point of $g(\cdot)$ as stated in the follow theorem.

Theorem 5 *Suppose that* $\{f_{N_k}(x_i^k)\}_{i=0}^{n_k}$, *with* $n_k > 1$, *satisfies* $f_{N_k}(x_{i+1}^k) - f_{N_k}^* = \theta_{N_k}(f_{N_k}(x_i^k) - f_{N_k}^*)$ *for all* $i = 0, 1, 2, ...$ *and some rate of convergence coefficient* $\theta_{N_k} \in (0, 1)$. *Then,* $\theta_{N_k} = g(\theta_{N_k})$.

Proof: See Appendix. □

In view of Theorems 4 and 5, we see that Subroutine B converges to a fixed point of $g(\cdot)$ and that the true rate of convergence coefficient is a fixed point of $g(\cdot)$ under the assumption of exact linear rate of convergence. Unfortunately, there may be more than one fixed point of $g(\cdot)$ and, hence, we cannot guarantee that Subroutine B converges to the rate of convergence coefficient from an arbitrary starting point. For example, if $n_k = 20$, $\theta = 0.15$, and $f_{N_k}(x_i^k) = \theta^i$, $i = 1, 2, , ..., n_k$, with $f_{N_k}(x_0^k) = 1$, then Subroutine B converges to the correct value 0.15 if initialized with $\hat{\theta}_k \in (0, 0.6633]$ and it converges to the incorrect value 0.8625 if initialized with $\hat{\theta}_k \in [0.6633, 1)$. (Here numbers are rounded to four digits.) The next theorem shows that Subroutine B indeed converges to the rate of convergence coefficient if initialized sufficiently close to that number for a wide range of values of θ_{N_k}. In our numerical tests, we find that the range of sufficiently close starting points is typically rather wide as in the example given above. This experience appears consistent with that of [12].

Theorem 6 *Suppose that* $\{f_{N_k}(x_i^k)\}_{i=0}^{n_k}$, *with* $n_k > 1$, *satisfies* $f_{N_k}(x_{i+1}^k) - f_{N_k}^* = \theta_{N_k}(f_{N_k}(x_i^k) - f_{N_k}^*)$ *for all* $i = 0, 1, 2, ...$ *and some rate of convergence coefficient* $\theta_{N_k} \in (0, 0.99]$. *If Subroutine B has generated the sequence* $\{a_j\}_{j=0}^{\infty}$, *ignoring the stopping criterion in Step 3, with* $a_0 = \hat{\theta}_k$ *sufficiently close to* θ_{N_k}, *then* $a_j \to \theta_{N_k}$, *as* $j \to \infty$.

Proof: See Appendix. □

It appears that Theorem 6 also holds for $\theta_{N_k} \in (0.99, 1)$. However, the verification of this requires a large computational effort as can be deduced from the proof of Theorem 6, which we have not carried out.

In view of Theorems 4, 5, and 6, we see that Subroutine B terminates in finite time under weak assumptions and it obtains the correct rate of convergence coefficient under somewhat stronger assumptions.

We next present a subroutine for estimating f^* based on a weighted average of estimates of $f_{N_l}^*$, $l = 1, 2, ..., k$. We let $\hat{f}_k^*$ and $\hat{\theta}_{k+1}$ denote the estimates of f^* and θ, respectively, available prior to the execution of Subroutine C.

Subroutine C (Computes estimate $\hat{f}_{k+1}^*$ of the optimal value f^*)

Input. Previous optimal value estimate $\hat{f}_k^*$, estimate of rate of convergence coefficient $\hat{\theta}_{k+1}$, and function values from the current stage $\{f_{N_k}(x_i^k)\}_{i=0}^{n_k}$.

Step 1. Compute an estimate of $f_{N_k}^*$:

$$\hat{m}_k := \min_{i=0,1,...,n_k-1} \frac{f_{N_k}(x_{n_k}^k) - \hat{\theta}_{k+1}^{n_k-i} f_{N_k}(x_i^k)}{1 - \hat{\theta}_{k+1}^{n_k-i}}. \tag{43}$$

Step 2. Compute

$$\hat{f}_{k+1}^* := \frac{N_k}{\sum_{l=1}^k N_l} \hat{m}_k + \frac{\sum_{l=1}^{k-1} N_l}{\sum_{l=1}^k N_l} \hat{f}_k^*. \tag{44}$$

Output. Optimal value estimate $\hat{f}_{k+1}^*$.

Step 1 of Subroutine C is the same as in [12] and in view of Lemma 1 provides a lower bound on $f_{N_k}^*$. The next result shows that $\hat{f}_{k+1}^*$, on average, is a lower bound on f^* under certain assumptions. (Similar lower bounds on the optimal value are determined in [28, 24].) Using a lower bound, we tend to conservatively estimate the computational effort needed to reach a near-optimal solution of P.

Proposition 3 *Suppose that* $\{f_{N_k}(x_i^k)\}_{i=0}^{n_k}$ *satisfies* $f_{N_k}(x_{i+1}^k) - f_{N_k}^* \leq \theta_{N_k}(f_{N_k}(x_i^k) - f_{N_k}^*)$ *for all* $i = 0, 1, 2, ...$ *and some rate of convergence coefficient* $\theta_{N_k} \in (0, 1)$. *If Subroutine C's input* $\hat{f}_k^* \leq f^*$ *and* $\hat{\theta}_{k+1} \geq \theta_{N_k}$, *then* $E[\hat{f}_{k+1}^*] \leq f^*$, *where* E *denotes the expectation with respect to the random sample of stage* k.

Proof: We deduce from Lemma 1 that $\hat{m}_k \leq f^*_{N_k}$ a.s. Hence, using the fact that $E[f^*_N] \leq f^*$ for all $N \in \mathbb{N}$, see, e.g., [24], we obtain that

$$E[\hat{f}^*_{k+1}] \leq E\left[\frac{N_k}{\sum_{l=1}^k N_l} f^*_{N_k} + \frac{\sum_{l=1}^{k-1} N_l}{\sum_{l=1}^k N_l} \hat{f}^*_k\right] = \frac{N_k}{\sum_{l=1}^k N_l} E[f^*_{N_k}] + \frac{\sum_{l=1}^{k-1} N_l}{\sum_{l=1}^k N_l} \hat{f}^*_k \leq f^*. \tag{45}$$

□

Under stronger assumptions, we also determine the asymptotic distribution of $\hat{f}^*_{k+1}$.

Proposition 4 *Suppose that Assumptions 3 and 4 hold, that* **P** *has a unique optimal solution* $x^* \in X$*, and that* $\{f_{N_l}(x^l_i)\}_{i=0}^{n_l}$*,* $l = 1, 2, ..., k$*, satisfy* $f_{N_l}(x^l_{i+1}) - f^*_{N_l} = \theta_{N_l}(f_{N_l}(x^l_i) - f^*_{N_l})$ *for some rate of convergence coefficients* $\theta_{N_l} \in (0, 1)$ *and for all* $i = 0, 1, 2, ..., n_l - 1$ *and* $l = 1, 2, ..., k$*. Let* $S_k = \sum_{l=1}^k N_l$*. If Subroutine C is applied at stages* $l = 1, 2, ..., k$ *with inputs* $\hat{f}_l$ *from the previous stage and* $\hat{\theta}_{l+1} \geq \theta_{N_l}$*, then* $S_k^{1/2}(\hat{f}^*_{k+1} - f^*) \Rightarrow \mathcal{N}(0, \sigma^2(x^*))$*, as* $S_k \to \infty$*.*

Proof: See Appendix. □

In view of Proposition 4, $\hat{f}^*_{k+1}$ is approximately normally distributed with mean f^* and variance $\sigma^2(x^*)/\sum_{l=1}^k N_l$ for large sample sizes under the stated assumptions.

The next subroutine estimates the function value at the end of stage k.

Subroutine D (Computes estimate $f_{N^*}(x^k_{n_k})$ of $f(x^k_{n_k})$)

Input. Verification sample size N^* and last iterate $x^k_{n_k}$.

Step 1. Generate an independent sample $\{\omega^*_j\}_{j=1}^{N^*}$ from $\mathbb{P}$.

Step 2. Compute the sample average

$$f_{N^*}(x^k_{n_k}) = \frac{1}{N^*} \sum_{j=1}^{N^*} F(x^k_{n_k}, \omega^*_j). \tag{46}$$

Output. Function value estimate $f_{N^*}(x^k_{n_k})$.

Subroutine D uses the standard sample average estimator to estimate $f(x^k_{n_k})$. Under Assumption 3, the central limit theorem states that for a given $x^k_{n_k} \in \mathbb{R}^d$, $f_{N^*}(x^k_{n_k})$ is approximately normally distributed with mean $f(x^k_{n_k})$ and variance $\sigma^2(x^k_{n_k})/N^*$ for large N^*.

The next subroutine deals with the computational work parameters.

Subroutine E (Computes estimates of computational work parameters w and w^*)

Input. Time t_k required to compute iterates during stage k and time t^*_k to verify the last function value of stage k as well as corresponding sample size N_k, iteration number n_k, and verification sample size N^*.

Step 1. Set $p_w = t_k/(N_k n_k)$ and $p_w^* = t_k^*/N^*$.

Output. Estimated computational work parameters p_w and p_w^*.

Subroutine E estimates the computational work parameters w and w^* in the computational work model (33) using two computing times observed during stage k. In principle, one could use past stage's computing times as well, but the simple Subroutine E performs well in the present context with the estimated computational work parameters p_w and p_w^* varying little from stage to stage in numerical tests.

Subroutines C and D do not guarantee that $\hat{f}_{k+1}^* \leq f_{N^*}(x_{n_k}^k)$, i.e., that the optimal value estimate is no larger than the estimated current objective function value. That inequality may be violated in early stages when estimates of f^* could be poor. These estimates are intended to be used in **S-SSCP**$_{k+1}$ and an estimated current function value that is within ϵ of the estimated optimal value would result in a trivial instance of **S-SSCP**$_{k+1}$: the optimal number of iterations for stage $k+1$ would be zero since the terminal surrogate state is already reached. To avoid to some extent such trivial instances of **S-SSCP**$_{k+1}$ prematurely, we adopt the following subroutine that makes adjustments to the estimates when needed.

Subroutine F (Sets estimates p_f and p^*, and gives surrogate optimality status)

Input. Estimates $\hat{f}_{k+1}^*$, $f_{N^*}(x_{n_k}^k)$, and p_σ, verification sample size N^*, total sample size $\sum_{l=1}^k N_l$, and stopping tolerance ϵ.

Step 1. If $\hat{f}_{k+1}^* + \epsilon < f_{N^*}(x_{n_k}^k)$, then set $p_f = f_{N^*}(x_{n_k}^k)$ and $p^* = \hat{f}_{k+1}^*$, and surrogate optimality status to "suboptimal."

Else set $p_f = f_{N^*}(x_{n_k}^k) + p_\sigma/\sqrt{N^*}$ and $p^* = \hat{f}_{k+1}^* - p_\sigma/\sqrt{\sum_{l=1}^k N_l}$. If $p^* + \epsilon < p_f$, set surrogate optimality status to "suboptimal." Otherwise set surrogate optimality status to "optimal."

Output. Surrogate optimality status and parameter estimates p_f and p^*.

Subroutine F sets $p_f = f_{N^*}(x_{n_k}^k)$ and $p^* = \hat{f}_{k+1}^*$, when the estimates $f_{N^*}(x_{n_k}^k)$ and $\hat{f}_{k+1}^*$ appear "reasonable" in the sense that the current estimates predict that a terminal surrogate state is not reached. In contrast, if $\hat{f}_{k+1}^* + \epsilon \geq f_{N^*}(x_{n_k}^k)$, i.e., a near-optimal solution appears to be reached, then Subroutine F replaces the estimates $f_{N^*}(x_{n_k}^k)$ and $\hat{f}_{k+1}^*$ by more conservative estimates. Specifically, $p_\sigma/\sqrt{N^*}$ is added to $f_{N^*}(x_{n_k}^k)$ and $p_\sigma/\sqrt{\sum_{l=1}^k N_l}$ is subtract off $\hat{f}_{k+1}^*$, which both represent shifting one standard deviation in the respective directions; see Proposition 4 and the discussion after Subroutine D. If either the original parameter estimates or the conservative ones predict that a near-optimal solution is not reached, we label the current solution "suboptimal" according to the surrogate model. Of course, a truly near-optimal solution may be labeled "suboptimal" due to the

uncertainty and approximations in the surrogate model. If both the original and the conservative estimates predict a near-optimal solution, we label the situation "optimal." Again, we stress that this does not imply that the current solution is nearly optimal. It merely indicates that the surrogate model has reached the terminal surrogate state and can therefore not be used to generate a sample size and a number of iterations for the next stage. In this case, as we see in the statement of Algorithm 2 below, we resort to a default policy for determining sample size and number of iterations.

4.2 Initialization Subroutine

The final subroutine determines parameters for **S-SSCP**$_1$, the first surrogate sample-size control problem to be solved at the beginning of Stage 1 of Algorithm 2.

Subroutine 0 (Computes initial parameter estimates p_f, p^*, and p_σ)

Input. Initial sample size N_0 and initial iterate x_0^0.

Step 1. Generate an independent sample $\{\omega_j^0\}_{j=1}^{N_0}$ from $\mathbb{P}$.

Step 2. Compute the sample average

$$f_{N_0}(x_0^0) = \frac{1}{N_0}\sum_{j=1}^{N_0} F(x_0^0, \omega_j^0) \tag{47}$$

and corresponding variance estimate

$$\hat{\sigma}_1^2 = \frac{1}{N_0 - 1}\sum_{j=1}^{N_0}(F(x_0^0, \omega_j^0) - f_{N_0}(x_0^0))^2. \tag{48}$$

Step 3. Set $p_f = f_{N_0}(x_0^0) + \hat{\sigma}_1/\sqrt{N_0}$, $p^* = \min\{0, f_{N_0}(x_0^0) - 1\}$, and $p_\sigma = \hat{\sigma}_1$.

Output. Parameter estimates p_f, p^*, and p_σ.

Step 3 of Subroutine 0 computes a likely conservative estimate of $f(x_0^0)$ by adding one standard deviation to the unbiased estimate $f_{N_0}(x_0^0)$. Step 3 also computes a rudimentary estimate of f^*. If problem specific information is available, the initial estimate of f^* may be improved.

4.3 Full Algorithm

Combining **S-SSCP**$_k$ and the above subroutines with Algorithm 1, we obtain Algorithm 2. Below, we indicate in parenthesis after a subroutine name the input parameters used in that subroutine.

Algorithm 2 (Adaptive Algorithm for P)

Data. Optimality tolerance $\epsilon > 0$; initial sample size $N_0 \in \mathbb{N}$; verification sample size N^*; default sample size factor $\gamma_N > 0$; default iteration number $\gamma_n \in \mathbb{N}$; smoothing parameter $\alpha_\theta \in [0, 1]$; initial estimate of rate of convergence coefficient $\hat{\theta}_1$; initial solution $x_0^0 \in X$; initial estimate of work coefficients p_w and p_w^*.

Step 0. Run **Subroutine 0**(N_0, x_0^0) to obtain p_f, p^*, and p_σ. Set $p_\theta = \hat{\theta}_1$, $x_0^1 = x_0^0$, and stage counter $k = 1$.

Step 1a. Solve **S-SSCP**$_k(p_f, p^*, p_\theta, p_\sigma, p_w, p_w^*, \epsilon)$ to obtain N_k and n_k.

Step 1b. Generate an independent sample $\{\omega_j^k\}_{j=1}^{N_k}$ from $\mathbb{P}$.

Step 2. For $i = 0$ to $n_k - 1$: Compute $x_{i+1}^k = A_{N_k}(x_i^k)$ using the sample generated in Step 1b. Let t_k denote the time to compute these iterates.

Step 3a. Run **Subroutine A**$(x_{n_k}^k, \{\omega_j^k\}_{j=1}^{N_k})$ to obtain p_σ.

Step 3b. Run **Subroutine B**$(\hat{\theta}_k, \{f_{N_k}(x_i^k)\}_{i=0}^{n_k})$ to obtain $\hat{\theta}$, and set $\hat{\theta}_{k+1} = \alpha_\theta \hat{\theta} + (1 - \alpha_\theta)\hat{\theta}_k$ and $p_\theta = \hat{\theta}_{k+1}$.

Step 3c. Run **Subroutine C**$(\hat{f}_k^*, \hat{\theta}_{k+1}, \{f_{N_k}(x_i^k)\}_{i=0}^{n_k})$ to obtain $\hat{f}_{k+1}^*$.

Step 3d. Run **Subroutine D**$(N^*, x_{n_k}^k)$ to obtain $f_{N^*}(x_{n_k}^k)$. Let t_k^* be the time required to run this subroutine.

Step 3e. Run **Subroutine E**$(t_k, t_k^*, N_k, n_k, N^*)$ to obtain p_w and p_w^*.

Step 3f. Run **Subroutine F**$(\hat{f}_{k+1}^*, f_{N^*}(x_{n_k}^k), p_\sigma, N^*, \sum_{l=1}^k N_l, \epsilon)$ to obtain surrogate optimality status and parameter estimates p_f and p^*. Set $x_0^{k+1} = x_{n_k}^k$, replace k by $k + 1$.

If surrogate optimality status is "suboptimal," then go to **Step 1a**.

Else (surrogate optimality status is "optimal"), set $N_k = \lceil \gamma_N N_{k-1} \rceil$ and $n_k = \gamma_n$ and go to **Step 1b**.

In Step 3b of Algorithm 2, the estimated rate of convergence coefficient is modified in view of previous estimates using exponential smoothing. Consequently, we avoid large fluctuations in this estimate. In Step 3f, Algorithm 2 resorts to a default policy defined by the parameters γ_N and γ_n when the surrogate sample-size control problem believes the current iterate satisfies the required tolerance.

Algorithm 2 is identical to Algorithm 1 except that the sample sizes and numbers of iterations are selected in a particular manner using **S-SSCP**$_k$. The underlying probability space $\bar{\Omega}$ of Algorithm 1 is also augmented with $\Omega^{N^*} \times \Omega^{N^*} \times \ldots$ for Algorithm 2 to account for the verification sample size; see Subroutine D. Since this change in probability space is trivial to account for in Theorem 1,

it follows that if the assumptions of that theorem are satisfied, then Algorithm 2 converges almost surely to a near-optimal solution. We note that it is straightforward to impose restrictions on the values of $\{N_k, n_k\}_{k=1}^{\infty}$ in **S-SSCP**$_k$ required by Theorem 1 through the construction of the sets D_N and D_n.

5 Computational Studies

In this section, we examine numerically the S-SSCP policy and compare it with practical alternatives, including the asymptotically optimal policy of the recent paper [30]. Specifically, we compare the computing time required to obtain a near-optimal solution by Algorithm 2 using different sample-size selection policies in Step 1a. As mentioned in Section 1, stochastic programs may also be solved by algorithms *not* based on SAA and VSAA. However, in this paper we do not compare across algorithmic frameworks and focus on efficient sample-size selection within VSAA when applied to smooth stochastic programs.

We implement Algorithm 2 in Matlab Version 7.4 and run the calculations on a laptop computer with 2.16 GHz processor, 2 GB RAM, and Windows XP operating system, unless otherwise stated. We use one iteration of the projected gradient method with Armijo step size rule (see, e.g., p. 67 of [33]) as the algorithm map $A_N(\cdot)$. The quadratic direction finding problem in the projected gradient method is solved using LSSOL [10] as implemented in TOMLAB 7.0 [14].

In all computational tests, we use parameters $\alpha = 0.5$ and $\beta = 0.8$ in Armijo step size rule (see p. 67 of [33]) as well as exponential smoothing parameter $\alpha_\theta = 1/3$ in Step 3b of Algorithm 2 and tolerance $\epsilon_\theta = 0.0001$ in Subroutine B. We use initial sample size $N_0 = 1000$, default sample size factor $\gamma_N = 1.1$, default iteration number $\gamma_n = 3$, and initial estimate of rate of convergence coefficient $\hat{\theta}_1 = 0.9$. Our initial computational work parameters p_w and p_w^* are 3 and 1, respectively.

5.1 Numerical Examples

We consider the following four problem instances. The first instance is a constructed example of **P** with known optimal solution. The second instance arises in investment portfolio optimization, the third in military and civilian search and rescue operations, and the fourth in engineering design with multiple performance functions. The second and fourth problem instances illustrate that Algorithm 2 may be used even if $F(\cdot, \omega)$ is nonsmooth, when proper approximations are used.

5.1.1 Problem Instance QUAD

Problem instance QUAD is defined in terms of

$$F(x, \omega) = \sum_{i=1}^{20} a_i (x_i - b_i \omega_i)^2 \tag{49}$$

with $b_i = 21 - i$, $i = 1, 2, ..., 20$, and $\omega = (\omega_1, \omega_2, ..., \omega_{20})'$ being a vector of 20 independent and $[0, 1]$-uniformly distributed random variables. We use the values $a_i = i$, $i = 1, 2, ..., 20$. (We have also examined other values for a_i and obtained similar results to those reported below.) The instances are unconstrained and we set X equal to a sufficiently large convex compact subset of $\mathbb{R}^{20}$ that includes all relevant solutions. Obviously, QUAD is strongly convex with a unique global minimizer $x^* = (x_1^*, ..., x_{20}^*)'$, where $x_i^* = b_i/2$. The optimal value is $\sum_{i=1}^{20} a_i b_i^2/12$. Even though solvable without VSAA, we use this simple problem instance to illustrate our approach. We set $x_0^0 = 0 \in \mathbb{R}^{20}$ and use relative optimality tolerance 0.001, i.e., $\epsilon = 0.001 p^*$ in Algorithm 2.

5.1.2 Problem Instance PORTFOLIO

The second problem instance, PORTFOLIO, is taken from [22] and arises in optimization of investment portfolios. We consider $d - 1$ financial instruments with random returns given by the $(d-1)$-dimensional random vector $\omega = \bar{R} + Qu$, where $\bar{R} = (\bar{R}_1, \bar{R}_2, ..., \bar{R}_{d-1})'$, with $\bar{R}_i$ being the expected return of instrument i, Q is an $(d-1)$-by-$(d-1)$ matrix, and u is a standard normal $(d-1)$-dimensional random vector. As in [22], we randomly generate $\bar{R}$ using an independent sample from a uniform distribution on $[0.9, 1.2]$ and Q using an independent sample from a uniform distribution on $[0, 0.1]$. The goal is to distribute one unit of wealth across the $d-1$ instruments such that the Conditional Value-at-Risk of the portfolio return is minimized and the expected portfolio return is no smaller than 1.05. We let $x_i \in \mathbb{R}$ denote the amount of investment in instrument i, $i = 1, 2, ..., d-1$. This results in the objective function (see [22, 36])

$$f(x) = \mathbb{E}\left[x_d + \frac{1}{1-t} \max\left\{ -\sum_{i=1}^{d-1} \omega_i x_i - x_d, 0 \right\} \right], \tag{50}$$

where $x = (x_1, x_2, ..., x_d)'$, with $x_d \in \mathbb{R}$ being an auxiliary decision variable, and $t \in (0, 1)$ is a probability level. The feasible region

$$X = \left\{ x \in \mathbb{R}^d \;\middle|\; \sum_{i=1}^{d-1} x_i = 1, \sum_{i=1}^{d-1} \bar{R}_i x_i \geq 1.05, x_i \geq 0, i = 1, 2, ..., d-1 \right\}. \tag{51}$$

We use $d = 101$ and $t = 0.9$.

The expression inside the expectation in (50) is not continuously differentiable everywhere for $\mathbb{P}$-almost every $\omega \in \Omega$. We overcome this difficulty by smoothing that expression using exponential smoothing with smoothing parameter 10^3; see [1, 52, 37, 31] for other applications of this approach as well as associated theory. This results in an error in function evaluation due to smoothing of less than $7 \cdot 10^{-4}$ for all $x \in \mathbb{R}^{101}$ and $\omega \in \mathbb{R}^{100}$. This problem instance illustrates that also nonsmooth problems may be solved approximately by Algorithm 2. Of course, as pointed out in [22], this instance of **P** can be reformulated as a conic-quadratic programming problem and solved directly without the use of VSAA. Hence, this is a convenient test instance as we are able to verify using cvx [11] that the solutions obtained by Algorithm 2 are indeed nearly optimal.

We use initial solution $x_0^0 = (0, 0, ..., 0, 1, 0, 0...., 0, -1)'$, where the 65-th component equals 1. In our data, the 65-th instrument has the largest expected return. Hence, the initial solution is the one with the largest expected portfolio return. We also set $\epsilon = 0.05p^*$. We implement this problem instance in Matlab Version 7.9 and run the calculations on a laptop computer with 2.26 GHz processor, 3.5 GB RAM, and Windows XP operating system.

5.1.3 Problem Instance SEARCH

The next problem instance generalizes a classical problem arising in search and detection applications. Consider an area of interest divided into d cells. A stationary target is located in one of the cells. A priori information gives that the probability that the target is in cell i is p_i, $i = 1, 2, ..., d$, with $\sum_{i=1}^{d} p_i = 1$. The goal is to optimally allocate one time unit of search effort such that the probability of not detecting the target is minimized (see, e.g., p. 5-1 in [51]). We generalize this problem and consider a random search effectiveness in cell i per time unit. We let $x = (x_1, x_2, ..., x_d)' \in \mathbb{R}^d$, with x_i representing the number of time units allocated to cell i, and let $\omega = (\omega_1, \omega_2, ..., \omega_d)'$, with ω_i, $i = 1, 2, ..., d$, being independent lognormally distributed random variables (with parameters $\xi_i = 100u_i$ and $\lambda_i = 0$, where $u_i \in (0, 1)$ are given data generated by independent sampling from a uniform distribution) representing the random search effectiveness in cell i. Then, the probability of not detecting the target is $f(x) = \mathbb{E}[F(x, \omega)]$, where

$$F(x, \omega) = \sum_{i=1}^{d} p_i e^{-\omega_i x_i}. \tag{52}$$

The decision variables are constrained by $\sum_{i=1}^{d} x_i = 1$ and $x_i \geq 0$, $i = 1, 2, ..., d$. We consider $d = 100$ cells. This problem instance, referred to as SEARCH, is convex. We observe that the expectation in the objective function can be computed by (numerically) solving d one-dimensional integrals. However, our goal is to illustrate Algorithm 2, which is based on VSAA, so we do not pursue that avenue. For this problem instance, we use $x_0^0 = (1/100, ..., 1/100)' \in \mathbb{R}^{100}$ and use relative optimality tolerance 0.001, i.e., $\epsilon = 0.001p^*$ in Algorithm 2.

5.1.4 Problem Instance TRUSS

The last problem instance deals with the design of a truss structure with topology given in Figure 2. The truss is subject to a random load L in its mid-span. L is lognormally distributed with mean 100 kN and standard deviation 10 kN. Let S_i be the yield stress of member i. Members 1-7 have lognormally distributed yield stresses with means 100, 120, 180, 190, 200, 210, and 220 N/mm^2, respectively. Members 1 and 2 have standard deviation 5 N/mm^2 and members 3-7 have standard deviations 10 N/mm^2. The yield stresses of members 1 and 2 are correlated with correlation coefficients 0.8. However, their correlation coefficients with the other yield stresses are 0.5. Similarly, the yield stresses of members 3-7 are correlated with correlation coefficients 0.8,

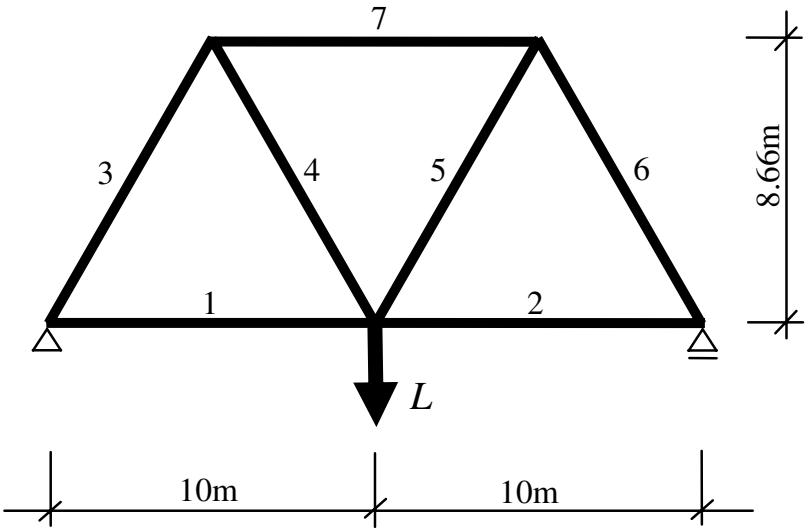

Figure 2: Design of Truss

but their correlation coefficients with the yield stresses of members 1 and 2 are 0.5. The load L is independent of the yield stresses.

The design vector $x = (x_1, x_2, ..., x_7)' \in \mathbb{R}^7$, where x_i is the cross-section area (in 1000 mm^2) of member i. The truss fails if any of the members exceed their yield stress and, hence, the probability of failure is $P[\bigcup_{i=1}^{7}\{S_i x_i - L/\zeta_i \leq 0\}]$, where $\zeta_i = 1/(2\sqrt{3})$ for $i = 1, 2$, and $\zeta_i = 1/\sqrt{3}$ for $i = 3, 4, ..., 7$ (see [41] for details). Using the approach in [41], see also [39], we find that this probability of failure can be approximated with high accuracy by

$$f(x) = \mathbb{E}[\max\{\rho, \max_{i=1,...,7}\{1 - \chi_8^2(r_i^2(x,\omega))\}\} \tag{53}$$

where $\rho > 0$ is an approximation parameter set equal to 20, $\chi_8^2(\cdot)$ is the Chi-square cumulative distribution function with 8 degrees of freedom, ω is an eight-dimensional random vector of independent standard normal random variables obtained from the original random variables $(L, S_1, ..., S_7)$ using a Nataf probability transformation, and $r_i(\cdot, \omega)$ is a smooth distance function. The function (53) is of form (1) and is continuously differentiable under moderate assumptions [39]. As in the case of PORTFOLIO, the expression inside the brackets in (53) is not continuously differentiable everywhere for $\mathbb{P}$-almost every $\omega \in \Omega$. We again overcome this difficulty by smoothing that expression using exponential smoothing with smoothing parameter 10^7; see [35]. This results in an error in function evaluation due to smoothing of less than $2 \cdot 10^{-7}$ for all $x \in \mathbb{R}^7$ and $\omega \in \Omega$. As the problem instance is not known to be convex, it illustrates that that the proposed approach may be effective

in such cases too. The goal in this design problem, denoted TRUSS, is to minimize $f(x)$ subject to $\sum_{i=1}^{7} x_i = 3$, $x_i \leq 0.5$, $x_i \geq 0.2$, $i = 1, 2, ..., 7$. We use $x_0^0 = (3/7, ..., 3/7)' \in \mathbb{R}^7$ and $\epsilon = 0.05p^*$.

5.2 Computational Results

We apply Algorithm 2 with different sample-size selection policies to the four problem instances. The measure of performance of a policy is the computing time in Algorithm 2 until the first time a stage k satisfies

$$\left[0, \max\left\{f_{N^*}(x_{n_k}^k) - \hat{f}_{k+1}^* + z_{1-\alpha_s} p_\sigma \sqrt{\frac{1}{N^*} + \frac{1}{\sum_{l=1}^{k} N_l}}, 0\right\}\right] \subset [0, \epsilon] \tag{54}$$

at the end of the stage. Here, $\epsilon > 0$ is the required tolerance and $z_{1-\alpha_s}$ is the $1-\alpha_s$ quantile of the standard normal cumulative distribution function. We use $\alpha_s = 0.05$ in our tests. The left-hand side in (54) is motivated as follows.

For a given $x \in \mathbb{R}^d$, $f_{N^*}(x)$ and $\hat{f}_{k+1}^*$ are independent when the sample used to compute $f_{N^*}(x)$ is independent of those used to compute $\hat{f}_{k+1}^*$. Hence, in view of Proposition 4 and the discussion after Subroutine D, $f_{N^*}(x) - \hat{f}_{k+1}^*$ is approximately normally distributed with mean $f(x) - f^*$ and variance $\sigma^2(x)/N^* + \sigma^2(x^*)/\sum_{l=1}^{k} N_l$ for large N^* and $\sum_{l=1}^{k} N_l$. Consequently, $[0, \max\{f_{N^*}(x) - \hat{f}_{k+1}^* + z_{1-\alpha_s}\sqrt{\sigma^2(x)/N^* + \sigma^2(x^*)/\sum_{l=1}^{k} N_l}, 0\}]$ is an approximate $100(1-\alpha_s)\%$ confidence interval for $f(x) - f^*$. We include the max-operator in the expression for this interval as $f(x) - f^* \geq 0$ for any $x \in X$. The assumptions underlying this confidence interval are not fully satisfied in the context of Algorithm 2 for three reasons. First, $\sigma(x)$ and $\sigma(x^*)$ are assumed known, which may not be the case. Second, since we are interested in $x = x_{n_k}^k$, the final iterate of the k-th stage, and $x_{n_k}^k$ is generated using the same samples as those underlying $\hat{f}_{k+1}^*$, $f_{N^*}(x_{n_k}^k)$ and $\hat{f}_{k+1}^*$ may not be independent. Third, we check the confidence interval after the completion of each stage in Algorithm 2, which implies sequential testing that may introduce a bias not accounted for in the confidence interval. In spite of these facts, we heuristically adopt the confidence interval with $\sigma(x)$ and $\sigma(x^*)$ replaced by the standard deviation estimate p_σ as the basis for our stopping criterion, which leads to (54). Consequently, we cannot guarantee that the left-hand side in (54) attains the stipulated coverage probability. However, we find empirically that the coverage probabilities are satisfactory. Specifically, Algorithm 2 stops, using (54), with an $x_{n_k}^k$ that fails to satisfy $f(x_{n_k}^k) - f^* \leq \epsilon$ in only 1% of 320 independent runs on QUAD, which is well within the 5% indicated by the confidence level 0.95. While not tested as comprehensively, the stopping criterion performs well also on the other problem instances. On PORTFOLIO, the stopping criterion never stops prematurely on 82 runs. It stops prematurely 1 time out of 90 and 0 times out of 90 for SEARCH and TRUSS, respectively, also well within the requested 5%.

The proximity to optimality of a solution obtained by Algorithm 2 could also be estimated using an optimality function [37], a hypothesis test of Karush-Kuhn-Tucker conditions [47], or an

Policy					QUAD		PORTFOLIO		SEARCH		TRUSS	
Name	s	d_N	d_n	d_f	avg.	st.d.	avg.	st.d.	avg.	st.d.	avg.	st.d.
S-SSCP high	10	40	20	30	372	229	33	11	393	173	575	257
S-SSCP medium	10	40	20	15	104	72	10	3	117	47	106	79
S-SSCP low	5	20	10	15	12	3	4	1	22	7	24	11

Table 1: Total computing times (seconds), over all stages, to solve **S-SSCP**$_k$ in Step 1a of Algorithm 2 with stopping criterion (54) for different values of s, d_N, d_n, and d_f. The times in columns 6, 8, 10, and 12 are averages over ten runs of Algorithm 2 when applied to QUAD, PORTFOLIO, SEARCH, and TRUSS, respectively. Standard deviations across the ten runs are listed in columns 7, 9, 11, and 13.

optimality gap estimate based on replications [28, 24]. However, we do not pursue those avenues here as (54) appears sufficient for our purpose of compare different sample-size selection policies.

In view of (54) and the fact that $\alpha_s = 0.05$, we select N^* so that the variability in $f_{N^*}(x^k_{n_k})$ is relatively small. Hence, we set $N^* = \lceil(\hat{\sigma}_1 z_{0.95}/(\epsilon/2))^2\rceil$, which is the smallest sample size that ensures that (54) is satisfied when $f_{N^*}(x^k_{n_k}) - \hat{f}^*_{k+1} = \epsilon/2$ and there is no uncertainty in f^*_{k+1}, i.e., $\sum_{l=1}^k N_l$ "equals" infinity.

We start by examining the computational effort required to solve **S-SSCP**$_k$ in Step 1a of Algorithm 2. As discussed in Subsection 3.3, that computational effort depends on the number of stages s, number possible sample sizes d_N, number of possible stage durations d_n, and number of discrete states d_f considered in **S-SSCP**$_k$. Table 1 gives total computing times in seconds to solve **S-SSCP**$_k$, $k = 1, 2, ..., k^*_\epsilon$, in Algorithm 2, where k^*_ϵ is the first stage satisfying the stopping criterion (54), for different values of s, d_N, d_n, and d_f when applied to QUAD (columns 6-7), PORTFOLIO (columns 8-9), SEARCH (columns 10-11), and TRUSS (columns 12-13). Termination occurs typically with k^*_ϵ between 5 and 15. The total computing times are averages over 10 independent runs and given in columns 6, 8, 10, and 12, with corresponding standard deviations given in columns 7, 9, 11, and 13. (The shorter times for PORTFOLIO are, in part, due to a faster computer.) Row 3 presents results for "high" values of s, d_N, d_n, and d_f as specified in columns 2-5 and we find the corresponding total computing times to be relatively long. Row 4 gives total computing times for a "medium" case with fewer discrete states d_f, which results in a reduction in the total computing time with a factor of about four. Row 5 considers the case with "low" values of s, d_N, d_n, and d_f. In this case, the total computing times for solving **S-SSCP**$_k$, $k = 1, 2, ..., k^*_\epsilon$, is reduced with a factor of approximately 10-30 and amounts to only about one second per solution of the surrogate sample-size control problem. The results for all problem instances correspond closely with what is predicted by the complexity result $O(sd_N d_n d_f^2)$ for **S-SSCP**$_k$; see Subsection 3.3. We next examine how different values of s, d_N, d_n, and d_f influence the quality of the resulting S-SSCP policy.

Using the same 10 independent runs of Algorithm 2 as in Table 1, rows 3-5 of Table 2 give

the average computing times in seconds over the 10 runs for Algorithm 2 applied to QUAD and PORTFOLIO, excluding the time of Step 1a to solve **S-SSCP**$_k$. (The remaining rows of Table 2 are discussed below.) Standard deviations of the times are given in columns 5 and 7. We find only a moderate variability across rows 3-5 in Table 2 and no clear indication that larger values of s, d_N, d_n, and d_f yield significantly better policies than smaller values.

Table 3 presents similar results as Table 2, but for problem instances SEARCH and TRUSS. Using the same 10 independent runs of Algorithm 2 as in Table 1, rows 3-5 of Table 3 present the average computing times in seconds over the 10 runs for Algorithm 2 excluding the time of Step 1a to solve **S-SSCP**$_k$. Again, there is only a moderate variability in computing times across rows 3-5 in Table 3 and no clear indication that larger values of s, d_N, d_n, and d_f yield significantly better policies than smaller values. The results of rows 3-5 in Tables 2 and 3 as well as those of tests with other values of s, d_N, d_n, and d_f not reported here, indicate that relatively small values of s, d_N, d_n, and d_f may be sufficient to generate reasonable S-SSCP policies. In view of Table 1, small values of s, d_N, d_n, and d_f result in essentially negligible computing times for **S-SSCP**$_k$. We anticipate that in real-world application of Algorithm 2 the main iterations of Step 2 in Algorithm 2 would require a significant amount of time, much more than the times for the present problem instances, due to large number of iterations, large sample sizes, and/or expansive function and gradient evaluations. In contrast, the total computing times to solve **S-SSCP**$_k$ would remain about the same as they are essentially independent of the application. They only depend on the values of s, d_N, d_n, and d_f. Hence, we conjecture that the time to obtain the S-SSCP policy is negligible in many applications.

We next compare the S-SSCP policy with other reasonable alternatives. Rows 6-8 of Table 2 give times on QUAD and PORTFOLIO for an "additive policy" where $N_1 = N^*/1000$ and $N_k = N_1 + (N^* - N_1)k/20$, $k = 2, 3, ...$, with $n_k = 5, 10$, and 20, respectively, rows 9-11 give times for a "multiplicative policy" where $N_1 = N^*/1000$ and $N_k = 1.5^{k-1}N_1$, $k = 2, 3, ...$, with $n_k = 5, 10$, and 20, respectively, and rows 12-14 give times for the same multiplicative policy as the previous rows except that $N_k = 2^{k-1}N_1$, $k = 2, 3, ...$ Rows 15-17 report results for a fixed policy with $N_k = N^*/2$ and $n_k = 5$ for all k. In Table 2 and elsewhere we indicate in brackets the number of terminations within a time limit, here one hour, when less than 10. Rows 18-29 follow policies deduced from the recommendation in [30]. Specifically, from an N_1 given in column 3, $N_k = 1.1^{k-1}N_1$, $k = 2, 3, ...$, for rows 18-23 and $N_k = 1.5^{k-1}N_1$, $k = 2, 3, ...$, for rows 24-29. The number of iterations carried out at each stage is determined adaptively. For QUAD, which is unconstrained, the stage ends when the estimated distance $\|(\nabla^2 f_{N_k}(x_i^k))^{-1}\nabla f_{N_k}(x_i^k)\|$ to an optimal solution is no greater than $K/\sqrt{N_k}$ as recommended in [30]. For constrained problems such as PORTFOLIO, no recommendation is given and we use the distance between the current solution and a minimizer over X of a quadratic model of $f_{N_k}(\cdot)$ at the current solution as an estimate of the distance to an optimal solution. Column 2 of rows 18-29 gives values of K. The policies of rows 18-29 are asymptotically optimal in a sense defined in [30]. We note that $N^*/1000$

Policy			QUAD		PORTFOLIO	
Name	n_k	N_1	avg.	st.d.	avg.	st.d.
S-SSCP high	adaptive	adaptive	201	137	382	119
S-SSCP medium	adaptive	adaptive	250	232	468	139
S-SSCP low	adaptive	adaptive	129	34	629	248
Additive	5	$10^{-3}N^*$	242	59	435	132
Additive	10	$10^{-3}N^*$	434	213	525	139
Additive	20	$10^{-3}N^*$	352	172	432	113
Mult. 1.5	5	$10^{-3}N^*$	631	267	1407	312
Mult. 1.5	10	$10^{-3}N^*$	666	379	1459	335
Mult. 1.5	20	$10^{-3}N^*$	759	460	1457	519
Mult. 2	5	$10^{-3}N^*$	494	233	1085	287
Mult. 2	10	$10^{-3}N^*$	867	426	989	204
Mult. 2	20	$10^{-3}N^*$	1080	1310	986	359
Fixed	5	$N^*/2$	-	[0]	2788	475
Fixed	10	$N^*/2$	-	[0]	2840	1752
Fixed	25	$N^*/2$	-	[0]	757	186
Mult. 1.1	$K = 1$	10	1434	170	-	[0]
Mult. 1.1	$K = 1$	100	1039	229	-	[0]
Mult. 1.1	$K = 1$	$10^{-3}N^*$	501	200	-	[0]
Mult. 1.1	$K = 0.1$	10	1521	318	-	[0]
Mult. 1.1	$K = 0.1$	100	1015	323	-	[0]
Mult. 1.1	$K = 0.1$	$10^{-3}N^*$	974	391	-	[0]
Mult. 1.5	$K = 1$	10	591	245	-	[0]
Mult. 1.5	$K = 1$	100	419	151	3600	[1]
Mult. 1.5	$K = 1$	$10^{-3}N^*$	305	79	2868	[1]
Mult. 1.5	$K = 0.1$	10	864	427	-	[0]
Mult. 1.5	$K = 0.1$	100	841	778	-	[0]
Mult. 1.5	$K = 0.1$	$10^{-3}N^*$	573	514	-	[0]

Table 2: Average and standard deviation of computing times (seconds) over ten runs of Algorithm 2 excluding the time of Step 1a when applied to QUAD and PORTFOLIO. Averages are only over runs completed within 3600 seconds. If less than 10 runs finished within that time limit, we report the number that did finish in brackets.

Policy			SEARCH		TRUSS	
Name	n_k	N_1	avg.	st.d.	avg.	st.d.
S-SSCP high	adaptive	adaptive	117	59	2157	1213
S-SSCP medium	adaptive	adaptive	141	62	1649	1290
S-SSCP low	adaptive	adaptive	118	45	2166	1793
Additive	5	$10^{-3}N^*$	134	74	1992	2240
Additive	10	$10^{-3}N^*$	253	158	2173	1369
Additive	20	$10^{-3}N^*$	210	120	6875	6280
Mult. 1.5	5	$10^{-3}N^*$	173	69	2661	2531
Mult. 1.5	10	$10^{-3}N^*$	204	145	2751	1181
Mult. 1.5	20	$10^{-3}N^*$	443	392	6756	7507
Mult. 2	5	$10^{-3}N^*$	122	67	3303	2805
Mult. 2	10	$10^{-3}N^*$	340	397	4557	4018
Mult. 2	20	$10^{-3}N^*$	404	389	10056	8098
Fixed	5	$N^*/2$	560	171	-	[0]
Fixed	10	$N^*/2$	942	353	-	[0]
Fixed	25	$N^*/2$	1138	438	-	[0]

Table 3: Average and standard deviation of computing times (seconds) over ten runs of Algorithm 2 excluding the time of Step 1a when applied to SEARCH and TRUSS. Averages are only over runs completed within 15000 seconds. If less than 10 runs finished within that time limit, we report the number that did finish in brackets.

is typically around 600 and 100 for QUAD and PORTFOLIO, respectively. In all cases, N_k is set to 3,000,000, if the above policies propose a sample size of larger than 3,000,000.

We see from Table 2 that the S-SSCP policies are often significantly better than the alternative ones for both problem instances. The first additive policy on QUAD (see row 6 of Table 2) appears to be reasonably efficient and competitive with S-SSCP medium. On PORTFOLIO, the three additive policies are competitive. Other alternative policies, however, may require as much as an order of magnitude more computing time. The alternative policies deduced from the recommendation in [30] (see rows 18-29 in Table 2) may result in poor computing times, especially for PORTFOLIO where only two instances solve within the time limit. It appears that the policies in [30] have a tendency to "over-solve" each stage. This follows as a consequence from the difficulty of accurately estimating the distance to an optimal solution.

Table 4 gives estimates of the value of $f(\cdot)$ at the final iterate of Algorithm 2 for the interesting cases in Table 2 using an independent sample of size $500,000$, which result in small estimator errors (95% confidence interval halfwidth of 0.8 and 0.002 for QUAD and PORTFOLIO, respectively). We report both average estimates over the 10 runs (columns 4 and 6) as well as standard deviations (columns 5 and 7). Since $f^* + \epsilon$ equals 1348.8 for QUAD and -0.335 for PORTFOLIO, Table 4 indicates that all the cases that terminated by (54) return solutions within the required tolerance. On PORTFOLIO, not all policies terminate within one hour, but we still report the estimated function values at that time.

Policy			QUAD		PORTFOLIO	
Name	n_k	N_1	avg.	st.d.	avg.	st.d.
S-SSCP high	adaptive	adaptive	1347.2	0.5	−0.348	0.002
S-SSCP medium	adaptive	adaptive	1347.5	0.4	−0.348	0.001
S-SSCP low	adaptive	adaptive	1347.5	0.2	−0.347	0.001
Additive	5	$10^{-3}N^*$	1347.9	0.5	−0.346	0.001
Additive	10	$10^{-3}N^*$	1347.6	0.5	−0.347	0.001
Additive	20	$10^{-3}N^*$	1347.6	0.3	−0.346	0.001
Mult. 1.5	5	$10^{-3}N^*$	1347.4	0.3	−0.346	0.001
Mult. 2	5	$10^{-3}N^*$	1347.7	0.5	−0.348	0.001
Mult. 1.1	$K = 1$	10	1347.4	0.4	−0.184	0.098
Mult. 1.1	$K = 1$	100	1347.5	0.3	−0.291	0.032
Mult. 1.1	$K = 1$	$10^{-3}N^*$	1347.8	0.5	−0.308	0.022
Mult. 1.1	$K = 0.1$	10	1347.7	0.5	0.127	0.428
Mult. 1.1	$K = 0.1$	100	1348.0	1.8	−0.266	0.052
Mult. 1.1	$K = 0.1$	$10^{-3}N^*$	1347.6	0.3	−0.308	0.019
Mult. 1.5	$K = 1$	10	1347.7	0.5	−0.285	0.074
Mult. 1.5	$K = 1$	100	1347.8	0.5	−0.307	0.057
Mult. 1.5	$K = 1$	$10^{-3}N^*$	1347.4	0.6	−0.332	0.027
Mult. 1.5	$K = 0.1$	10	1347.6	0.5	−0.097	0.351
Mult. 1.5	$K = 0.1$	100	1348.1	1.2	−0.325	0.026
Mult. 1.5	$K = 0.1$	$10^{-3}N^*$	1347.8	0.8	−0.323	0.039

Table 4: Average and standard deviation of estimated objective function values over ten runs of Algorithm 2 for QUAD and PORTFOLIO.

k	S-SSCP high		Additive		Mult. factor 2		Mult. factor 1.1		Mult. factor 1.5	
	N_k	n_k	N_k	n_k	N_k	n_k	N_k	n_k	N_k	n_k
1	100	37	586	5	608	10	100	45	621	55
2	10000	32	29881	5	1216	10	110	24	932	24
3	11000	9	59176	5	2432	10	121	22	1397	23
4	39995	4	88471	5	4864	10	133	22	2096	21
5	43995	3	117766	5	9728	10	146	9	3144	16
6	48395	3			19456	10	161	11	4716	12
7					38912	10	177	17	7074	18
8					77824	10	195	20	10610	14
9					155648	10	214	13	15916	13
10							236	14	23873	14
11							259	17	35810	19
12							285	12	53715	22
13							314	16	80572	25
14							345	19	120859	7
15							380	13		
...							...	...		
71							78975	14		

Table 5: Examples of sample sizes N_k and numbers of iterations n_k for QUAD in policies S-SSCP high (columns 2-3), Additive with $n_k = 5$ and $N_1 = N^*/1000$ (columns 4-5), Multiplicative with factor 2, $n_k = 10$, and $N_1 = N^*/1000$ (columns 6-7), Multiplicative with factor 1.1, $K = 1$, and $N_1 = 100$ (columns 8-9), and Multiplicative with factor 1.5, $K = 1$, and $N_1 = N^*/1000$ (columns 10-11).

Table 5 presents examples of sample sizes and number of iterations for QUAD for five runs using policies from Table 2. Columns 2-3 show that the policy S-SSCP high requires six stages and that the number of iterations tends to decrease as the stages progress. **S-SSCP**$_k$ identifies this as a computationally efficient approach as the later stages necessarily would require a large sample size. We note that in QUAD N^* is about $6 \cdot 10^5$. Columns 4-5 shows comparable results for the additive policy with $n_k = 5$ and $N_1 = N^*/1000$. The fixed increase in sample size after each stage appears to be to large in this case as the sample size reaches unnecessarily high values. The situation is similar for the multiplicative policy with factor 2, $n_k = 10$, and $N_1 = N^*/1000$; see columns 6-7. For the multiplicative policies with factor 1.1, $K = 1$, and $N_1 = 100$ (columns 8-9) and with factor 1.5, $K = 1$, and $N_1 = N^*/1000$ (columns 10-11), we find that the adaptive rule for determining the number of iterations for each stage tends to result in high values of n_k. Hence, the algorithm "over-solves" each stage, which may result in long computing times. Also, the number of stages may become excessive if the multiplicative factor is small; see columns 8-9. We observe similar behaviors to those in Table 5 for the other problem instances.

Returning to Table 3, we see that also in the case of SEARCH and TRUSS the alternative policies perform poorly. Here, we do not examine the policies from [30] due to their poor performance in Table 2. On SEARCH, the S-SSCP policies appear to be the fastest, but one additive policy (row 6) and one multiplicative policy (row 12) are competitive. On the problem instance TRUSS (see columns 6 and 7 of Table 3), the S-SSCP policies again outperform most alternative policies with two additive policies (rows 6 and 7) being competitive. The fastest additive policy on average, however, has a larger standard deviations (coefficient of variations of roughly 1.1) than that of the S-SSCP policies (coefficient of variation of 0.7 on average). Hence, the user of that additive policy is exposed to a significant risk of having a long computing time even with this "good" policy. We observe that the best alternative policy for SEARCH (see row 12 of Table 3) is only the fifth best alternative policy for TRUSS. Hence, as could be expected, a good alternative policy for one problem instance may not be particularly good for another. Of course, this makes the process of selecting a policy manually or by trial-and-error rather difficult.

Table 6 gives estimates of the value of $f(\cdot)$ at the final iterate of Algorithm 2 when applied to SEARCH and TRUSS, again using a sample of size $500,000$ (which gives 95% confidence interval halfwidth of 0.0002 and 0.0006 for SEARCH and TRUSS, respectively). We report both average estimates over the 10 runs (columns 4 and 6) as well as standard deviations (columns 5 and 7). Since $f^* + \epsilon$ equals 0.5619 for SEARCH and 0.0241 for TRUSS, Table 6 indicates that all the cases return solutions within the required tolerance.

In view of the above results, we see that even on simple problem instances a poor choice of sample-size selection policy may result in extremely long computing times. Moreover, the recommendations from [30], which are based on asymptotic analysis of sampling error and algorithmic progress, may not be helpful in practice. In fact, on the problem instances examined, these rec-

Policy			SEARCH		TRUSS	
Name	n_k	N_1	avg.	st.d.	avg.	st.d.
S-SSCP high	adaptive	adaptive	0.5614	0.0001	0.0229	0.0002
S-SSCP medium	adaptive	adaptive	0.5614	0.0002	0.0231	0.0003
S-SSCP low	adaptive	adaptive	0.5614	0.0001	0.0229	0.0004
Additive	5	$10^{-3}N^*$	0.5614	0.0001	0.0231	0.0004
Additive	10	$10^{-3}N^*$	0.5614	0.0001	0.0229	0.0002
Mult. 1.5	5	$10^{-3}N^*$	0.5614	0.0001	0.0231	0.0003
Mult. 1.5	10	$10^{-3}N^*$	0.5614	0.0001	0.0231	0.0002
Mult. 2	5	$10^{-3}N^*$	0.5614	0.0001	0.0231	0.0003
Mult. 2	10	$10^{-3}N^*$	0.5614	0.0001	0.0231	0.0003

Table 6: Average and standard deviation of objective function values over ten runs of Algorithm 2 for SEARCH and TRUSS.

ommendations perform worse than simple additive or multiplicative policies. On the other hand, the S-SSCP policy appears to be robust across values of the parameters s, d_N, d_n, and d_f and it performs well even on ill-conditioned problems not reported here. In contrast to rigid additive and multiplicative policies, the S-SSCP policy initially recommends many iterations per stage but reduces the number as the sample size increases in later stages. When the sample size is large and the surrogate terminal state is almost satisfied, the policy recommends a cautious increase in sample size.

6 Conclusions

We considered the solution of smooth stochastic programs by sample average approximations and formulated the problem of selecting efficient sample sizes as a discrete-time optimal-control problem that aims to minimize the expected computing time to reach a near-optimal solution. The optimal-control problem is intractable, but we approximate it by a surrogate sample-size control problem using state aggregation and the result of a novel model of algorithmic behavior. The surrogate sample-size control problem depends on unknown parameters that we estimate as the algorithm progresses. Hence, we solve the surrogate sample-size control problem repeatedly within a receding-horizon framework.

Even with estimates of parameters, the surrogate sample-size control problem provides a policy for selecting sample sizes and number of iterations that outperforms most plausible alternative policies including policies known to be optimal in some asymptotic sense. The surrogate sample-size control problem provides a policy that appears to be robust to changing characteristics of problem instances such as ill-conditioning. In comparison, the alternative policies may result in dramatically varying computing times. Of course, we do not examine all possible policies in this paper, among which there is likely to be some that are better than the surrogate sample-size

control policy. However, we illustrate the difficulty a user faces when selecting a policy prior to calculations. We also show that guidance provided by recommendations in the literature may not be helpful in practice. The approach derived in this paper eliminates the need for users to select a policy through extensive trial-and-error or guesswork and, hence, facilitates implementation of stochastic programming algorithms in decision-support tools.

Acknowledgement

This study is supported by AFOSR Young Investigator grant F1ATA08337G003. The author is grateful for valuable discussions with Roberto Szechtman, Naval Postgraduate School. The author also thanks Alexander Shapiro, Georgia Institute of Technology, for assistance with two technical results.

Appendix

This appendix includes proofs of results in Section 4.

Proof of Proposition 2. By assumption, for any $i = 0, 1, ..., n_k - 1$ and $a \in (0, 1)$

$$\frac{f_{N_k}(x^k_{n_k}) - a^{n_k - i} f_{N_k}(x^k_i)}{1 - a^{n_k - i}} < \frac{f_{N_k}(x^k_{n_k}) - a^{n_k - i} f_{N_k}(x^k_{n_k})}{1 - a^{n_k - i}} = f_{N_k}(x^k_{n_k}). \tag{55}$$

Hence, $\phi(a) < f_{N_k}(x^k_{n_k})$ and

$$f_{N_k}(x^k_i) - \phi(a) > 0 \tag{56}$$

for any $i = 0, 1, ..., n_k$ and $a \in (0, 1)$. Consequently, the logarithmic transformation of the data in Step 2 of Subroutine B is permissable when $a_j \in (0, 1)$ and regression coefficients $\log a_{j+1}$ and $\log b_{j+1}$, $j = 0, 1, ...$, are given by the standard linear least-square regression formulae. Specifically,

$$\log a_{j+1} = \frac{\sum_{i=0}^{n_k} (i - n_k/2)[\log(f_{N_k}(x^k_i) - \phi(a_j)) - \sum_{i'=0}^{n_k} \log(f_{N_k}(x^k_{i'}) - \phi(a_j))/(n_k + 1)]}{\sum_{i=0}^{n_k} (i - n_k/2)^2}. \tag{57}$$

Since the denominator in (57) simplifies to $(n_k^3 + 3n_k^2 + 2n_k)/12$, we obtain using the definition of $\alpha_i = 12(i - n_k/2)/(n_k^3 + 3n_k^2 + 2n_k)$ in Proposition 2 that

$$\begin{aligned}
a_{j+1} &= \exp\left\{\sum_{i=0}^{n_k} \alpha_i \left[\log(f_{N_k}(x^k_i) - \phi(a_j)) - \sum_{i'=0}^{n_k} \log(f_{N_k}(x^k_{i'}) - \phi(a_j))/(n_k + 1)\right]\right\} \\
&= \prod_{i=0}^{n_k} \exp\left\{\alpha_i \left[\log(f_{N_k}(x^k_i) - \phi(a_j)) - \sum_{i'=0}^{n_k} \log(f_{N_k}(x^k_{i'}) - \phi(a_j))/(n_k + 1)\right]\right\} \\
&= \prod_{i=0}^{n_k} \left((f_{N_k}(x^k_i) - \phi(a_j))^{\alpha_i} \prod_{i'=0}^{n_k} (f_{N_k}(x^k_{i'}) - \phi(a_j))^{-\alpha_i/(n_k+1)} \right) \\
&= \prod_{i=0}^{n_k} (f_{N_k}(x^k_i) - \phi(a_j))^{\alpha_i} \prod_{i'=0}^{n_k} (f_{N_k}(x^k_{i'}) - \phi(a_j))^{-\sum_{i=0}^{n_k} \alpha_i/(n_k+1)}.
\end{aligned} \tag{58}$$

We find that

$$\begin{aligned}\sum_{i=0}^{n_k}\alpha_i &= \sum_{i=0}^{n_k}12(i-n_k/2)/(n_k^3+3n_k^2+2n_k)\\ &= 12(n_k(n_k+1)/2-(n_k+1)n_k/2)/(n_k^3+3n_k^2+2n_k)=0 \end{aligned} \tag{59}$$

and consequently

$$a_{j+1}=\prod_{i=0}^{n_k}(f_{N_k}(x_i^k)-\phi(a_j))^{\alpha_i}. \tag{60}$$

By definition, $\alpha_i = 12(i-n_k/2)/(n_k^2+3n_k^2+2n_k) = -12(n_k-i-n_k/2)/(n_k^2+3n_k^2+2n_k) = -\alpha_{n_k-i}$. Hence,

$$\begin{aligned} a_{j+1} &= \prod_{i=0}^{n_k^0-1}(f_{N_k}(x_i^k)-\phi(a_j))^{\alpha_i}\prod_{i=n_k^0}^{n_k}(f_{N_k}(x_i^k)-\phi(a_j))^{\alpha_i}\\ &= \prod_{i=0}^{n_k^0-1}(f_{N_k}(x_i^k)-\phi(a_j))^{-\alpha_{n_k-i}}\prod_{i=n_k^0}^{n_k}(f_{N_k}(x_i^k)-\phi(a_j))^{\alpha_i}\\ &= \prod_{i'=n_k}^{n_k^0}(f_{N_k}(x_{n_k-i'}^k)-\phi(a_j))^{-\alpha_{i'}}\prod_{i=n_k^0}^{n_k}(f_{N_k}(x_i^k)-\phi(a_j))^{\alpha_i},\\ &= \prod_{i'=n_k^0}^{n_k}(f_{N_k}(x_{n_k-i'}^k)-\phi(a_j))^{-\alpha_{i'}}\prod_{i=n_k^0}^{n_k}(f_{N_k}(x_i^k)-\phi(a_j))^{\alpha_i}, \end{aligned}$$

where we use the fact that $\alpha_{n_k/2}=0$ when n_k is an even number. The expression for $g(\cdot)$ then follows by combining the two products. The positivity of $g(a_j)$ follows trivially from (56), as $g(a_j)$ is a product of positive numbers. Since $f_{N_k}(x_i^k) > f_{N_k}(x_{i+1}^k)$ for all $i=0,1,...,n_k-1$, $(f_{N_k}(x_i^k)-\phi(a))/(f_{N_k}(x_{n_k-i}^k)-\phi(a)) < 1$ for all $i=n_k^0,n_k^0+1,...,n_k$. Moreover, $\alpha_i>0$ for all $i=n_k^0,n_k^0+1,...,n_k$. Hence, it follows that $g(a)<1$. □

Proof of Theorem 4. We first show that the derivative $dg(a)/da$ exists and is positive on $(0,1)$. For any $a\in(0,1)$ and $i=n_k^0,n_k^0+1,...,n_k$, let $h_i:(0,1)\to\mathbb{R}$ be defined by

$$h_i(a) := \frac{f_{N_k}(x_i^k)-\phi(a)}{f_{N_k}(x_{n_k-i}^k)-\phi(a)}. \tag{61}$$

By (56), $h_i(a)>0$ for any $a\in(0,1)$ and $i=n_k^0,n_k^0+1,...,n_k$ and consequently

$$g(a)=\exp\left(\sum_{i=n_k^0}^{n_k}\alpha_i\log h_i(a)\right). \tag{62}$$

By straightforward derivation we obtain that

$$\begin{aligned}\frac{dg(a)}{da} &= g(a)\sum_{i=n_k^0}^{n_k}\alpha_i\frac{dh_i(a)/da}{h_i(a)} \\ &= g(a)\sum_{i=n_k^0}^{n_k}\alpha_i\frac{(f_{N_k}(x_i^k)-f_{N_k}(x_{n_k-i}^k))d\phi(a)/da}{(f_{N_k}(x_i^k)-\phi(a))(f_{N_k}(x_{n_k-i}^k)-\phi(a))},\end{aligned} \tag{63}$$

where

$$\frac{d\phi(a)}{da} = \frac{1}{n_k}\sum_{i=0}^{n_k-1}\frac{(n_k-i)a^{n_k-i-1}(f_{N_k}(x^k_{n_k}) - f_{N_k}(x^k_i))}{(1-a^{n_k-i})^2}. \tag{64}$$

Since $f_{N_k}(x^k_{n_k}) - f_{N_k}(x^k_i) < 0$ for all $i = 0, 1, ..., n_k - 1$ by assumption, it follows that $d\phi(a)/da < 0$ for all $a \in (0,1)$. Again, by assumption, $f_{N_k}(x^k_i) - f_{N_k}(x^k_{n_k-i}) < 0$ for all $i = n^0_k, n^0_k + 1, ..., n_k$. Hence, by (56) and the fact that $\alpha_i > 0$ for all $i = n^0_k, n^0_k + 1, ..., n_k$, we conclude that $dg(a)/da > 0$ for any $a \in (0,1)$

Since $\{a_j\}_{j=0}^{\infty}$ is contained in the compact set $[0,1]$ by Proposition 2, it follows that there exists a subsequence $\{a_j\}_{j\in J}$, with $J \subset \mathbb{N}$, and an $a^* \in [0,1]$ such that $a_j \to^J a^*$, as $j \to \infty$. By the mean value theorem, we obtain that for every $j = 0, 1, 2, ...,$, there exists an $s_j \in [0,1]$ such that

$$a_{j+2} - a_{j+1} = g(a_{j+1}) - g(a_j) = \frac{dg(a_j + s_j(a_{j+1} - a_j))}{da}(a_{j+1} - a_j). \tag{65}$$

Since $dg(a_j + s_j(a_{j+1} - a_j))/da > 0$, it follows that $\{a_j\}_{j=0}^{\infty}$ generated by Subroutine B initialized with $a_0 \in (0,1)$ is either strictly increasing or strictly decreasing. That is, if $a_0 < a_1$, then $a_j < a_{j+1}$ for all $j \in \mathbb{N}$. If $a_0 > a_1$, then $a_j > a_{j+1}$ for all $j \in \mathbb{N}$. Hence, $a_j \to a^*$ as $j \to \infty$. Similarly, $g(a_j) \in (0,1)$ by Proposition 2 and there must exists a convergent subsequence of $\{g(a_j)\}_{j=0}^{\infty}$ that convergence to a point $g^* \in [0,1]$. Since $a_{j+1} = g(a_j)$, $\{g(a_j)\}_{j=0}^{\infty}$ is either strictly increasing or strictly decreasing and therefore $g(a_j) \to g^*$, as $j \to \infty$. Since $a_{j+1} = g(a_j)$ for all $j \in \mathbb{N}$ and $a_j \to a^*$ and $g(a_j) \to g^*$, as $j \to \infty$, we have that $a^* = g^*$. By continuity of $g(\cdot)$ on $(0,1)$, if $a^* \in (0,1)$, then $g(a_j) \to g(a^*)$, as $j \to \infty$. Hence, $g(a^*) = g^* = a^*$. If $a^* = 0$, then $g^* = 0$. If $a^* = 1$, then $g^* = 1$. Since by definition $g(0) = 0$ and $g(1) = 1$, it follows that $a^* = g(a^*)$ in these two cases too. The finite termination of Subroutine B follows directly from the fact that $\{a_j\}_{j=0}^{\infty}$ converges. □

Proof of Theorem 5. Since $f_{N_k}(x^k_i) = f^*_{N_k} + (\theta_{N_k})^i(f_{N_k}(x^k_0) - f^*_{N_k})$ for all $i = 0, 1, 2, ...,$

$$\phi(\theta_{N_k}) = \frac{1}{n_k}\sum_{i=0}^{n_k-1}\frac{f_{N_k}(x^k_{n_k}) - (\theta_{N_k})^{n_k-i}f_{N_k}(x^k_i)}{1-(\theta_{N_k})^{n_k-i}} = f^*_{N_k}. \tag{66}$$

It then follows by (60) that

$$\begin{aligned} g(\theta_{N_k}) &= \prod_{i=0}^{n_k}(f_{N_k}(x^k_i) - f^*_{N_k})^{\alpha_i} \\ &= \prod_{i=0}^{n_k}(f^*_{N_k} + \theta^i_{N_k}(f_{N_k}(x^k_0) - f^*_{N_k}) - f^*_{N_k})^{\alpha_i} \\ &= \theta^{\sum_{i=0}^{n_k} i\alpha_i}_{N_k}(f_{N_k}(x^k_0) - f^*_{N_k})^{\sum_{i=0}^{n_k}\alpha_i}. \end{aligned}$$

Since $\sum_{i=0}^{n_k} i^2 = (n_k+1)(n_k^2/3 + n_k/6)$,

$$\begin{aligned} \sum_{i=0}^{n_k} i\alpha_i &= \sum_{i=0}^{n_k} 12(i^2 - in_k/2)/(n_k^3 + 3n_k^2 + 2n_k) \\ &= 12((n_k+1)(n_k^2/3 + n_k/6) - n_k(n_k+1)n_k/4)/(n_k^3 + 3n_k^2 + 2n_k) = 1. \end{aligned}$$

Since $\sum_{i=0}^{n_k} \alpha_i = 0$ by (59), the conclusion follows. □

Proof of Theorem 6. By Theorem 5 and (66), $\theta_{N_k} = g(\theta_{N_k})$ and $\phi(\theta_{N_k}) = f^*_{N_k}$. Consequently, it follows from (63) and the assumption of exact linear rate that

$$\frac{dg(\theta_{N_k})}{da} = \theta_{N_k} \sum_{i=n_k^0}^{n_k} \alpha_i \frac{(\theta^i_{N_k} - \theta^{n_k-i}_{N_k}) d\phi(\theta_{N_k})/da}{\theta^{n_k}_{N_k}(f_{N_k}(x^k_0) - f^*_{N_k})}. \tag{67}$$

Since

$$\begin{aligned} \frac{d\phi(\theta_{N_k})}{da} &= \frac{1}{n_k} \sum_{i=0}^{n_k-1} \frac{(n_k - i)\theta^{n_k-i-1}_{N_k}(\theta^{n_k}_{N_k} - \theta^i_{N_k})(f_{N_k}(x^k_0) - f^*_{N_k})}{(1 - \theta^{n_k-i}_{N_k})^2} \\ &= -\frac{1}{n_k} \sum_{i=0}^{n_k-1} \frac{(n_k - i)\theta^{n_k-1}_{N_k}(f_{N_k}(x^k_0) - f^*_{N_k})}{1 - \theta^{n_k-i}_{N_k}}, \end{aligned} \tag{68}$$

we obtain that

$$\begin{aligned} \frac{dg(\theta_{N_k})}{da} &= \left(-\frac{1}{n_k} \sum_{i=0}^{n_k-1} \frac{(n_k - i)\theta^{n_k-1}_{N_k}}{1 - \theta^{n_k-i}_{N_k}} \right) \theta_{N_k} \sum_{i=n_k^0}^{n_k} \alpha_i \frac{\theta^i_{N_k} - \theta^{n_k-i}_{N_k}}{\theta^{n_k}_{N_k}} \\ &= \left(\frac{1}{n_k} \sum_{i=0}^{n_k-1} \frac{n_k - i}{1 - \theta^{n_k-i}_{N_k}} \right) \sum_{i=n_k^0}^{n_k} \alpha_i (\theta^{n_k-i}_{N_k} - \theta^i_{N_k}) \\ &= \left(\frac{1}{n_k} \sum_{i'=1}^{n_k} \frac{i'}{1 - \theta^{i'}_{N_k}} \right) \sum_{i=n_k^0}^{n_k} \alpha_i (\theta^{n_k-i}_{N_k} - \theta^i_{N_k}). \end{aligned} \tag{69}$$

If n_k is even, then using (59) we find that

$$\begin{aligned} \sum_{i=n_k^0}^{n_k} \alpha_i &= 0 - \sum_{i=0}^{n_k/2} 12(i - n_k/2)/(n_k^3 + 3n_k^2 + 2n_k) \\ &= -12[(n_k/2)(n_k/2+1)/2 - (n_k/2+1)n_k/2]/(n_k^3 + 3n_k^2 + 2n_k) = (3/2)/(n_k+1). \end{aligned}$$

If n_k is odd, then using (59) we find that

$$\begin{aligned} \sum_{i=n_k^0}^{n_k} \alpha_i &= 0 - \sum_{i=0}^{n_k/2-1/2} 12(i - n_k/2)/(n_k^3 + 3n_k^2 + 2n_k) \\ &= -12[(n_k/2 - 1/2)(n_k/2 - 1/2 + 1)/2 - (n_k/2 - 1/2 + 1)n_k/2]/(n_k^3 + 3n_k^2 + 2n_k) \\ &= (3/2)(n_k+1)/(n_k(n_k+2)). \end{aligned}$$

Since $1/(n_k+1) < (n_k+1)/(n_k(n_k+2))$,

$$\sum_{i=n_k^0}^{n_k} \alpha_i \le (3/2)(n_k+1)/(n_k(n_k+2)) \tag{70}$$

for all $n_k = 2, 3, ...$

The first multiplicative term in (69) decomposes as follows:

$$\frac{1}{n_k}\sum_{i=1}^{n_k}\frac{i}{1-\theta_{N_k}^i}=\frac{n_k+1}{2}+\frac{1}{n_k}\sum_{i=1}^{n_k}\left(\frac{i}{1-\theta_{N_k}^i}-i\right)=\frac{n_k+1}{2}+\frac{1}{n_k}\sum_{i=1}^{n_k}\frac{i\theta_{N_k}^i}{1-\theta_{N_k}^i}. \tag{71}$$

Since $1/(1-\theta_{N_k}^{i'})\leq 1/(1-\theta_{N_k})$ for any $i'\in\mathbb{N}$, we obtain from (69) using (70) and (71) that

$$\frac{dg(\theta_{N_k})}{da} \leq \left(\frac{n_k+1}{2}+\frac{1}{n_k}\frac{1}{1-\theta_{N_k}}\sum_{i'=1}^{\infty}i'\theta_{N_k}^{i'}\right)\frac{3}{2}\frac{n_k+1}{n_k(n_k+2)}. \tag{72}$$

Using the mean of the geometric distribution, we deduce that $\sum_{i=1}^{\infty}i\theta_{N_k}^i=\theta_{N_k}/(1-\theta_{N_k})^2$. Hence,

$$\frac{dg(\theta_{N_k})}{da}\leq\left(\frac{n_k+1}{2}+\frac{1}{n_k}\frac{\theta_{N_k}}{(1-\theta_{N_k})^3}\right)\frac{3}{2}\frac{n_k+1}{n_k(n_k+2)}\leq\frac{3}{4}\frac{n_k+1}{n_k}+\frac{3}{2}\frac{1}{n_k^2}\frac{\theta_{N_k}}{(1-\theta_{N_k})^3}. \tag{73}$$

Consequently, if $n_k>(1+\sqrt{1+72\beta})/6$, where $\beta=\theta_{N_k}/(1-\theta_{N_k})^3$, then the right-hand size in (73) is less than one. Hence, for $n_k>(1+\sqrt{1+72\beta})/6$, $dg(\theta_{N_k})/da<1$. Since $\theta_{N_k}/(1-\theta_{N_k})^3\leq 0.99/(1-0.99)^3$ for all $\theta_{N_k}\in[0,0.99]$, it follows that when $n_k\geq 1408$, $dg(\theta_{N_k})/da<1$ for any $\theta_{N_k}\in[0,0.99]$. It then follows by the fixed point theorem that under the assumption that $n_k\geq 1408$, $a_j\to\theta_{N_k}$, as $j\to\infty$, whenever a_0 is sufficiently close to θ_{N_k}.

It appears difficult to examine $dg(\theta_{N_k})/da$ analytically for $2<n_k<1408$. However, we show that $dg(\theta_{N_k})/da<1$ for all $\theta_{N_k}\in(0,0.99]$ and $2\leq n_k<1408$ using the following numerical scheme. (We note that the case with $n_k=2$ is easily checked analytically, but we do not show that for brevity.) We consider the function $\gamma:[0,1)\to\mathbb{R}$ defined for any $\theta\in[0,1)$ by

$$\gamma(\theta):=\left(\frac{1}{n_k}\sum_{i'=1}^{n_k}\frac{i'}{1-\theta^{i'}}\right)\sum_{i=n_k^0}^{n_k}\alpha_i(\theta^{n_k-i}-\theta^i). \tag{74}$$

Obviously, for $\theta_{N_k}\in(0,1)$, $\gamma(\theta_{N_k})=dg(\theta_{N_k})/da$. Straightforward derivation yields that

$$\frac{d\gamma(\theta)}{d\theta}=\sum_{i'=1}^{n_k}\sum_{i=n_k^0}^{n_k}\frac{\alpha_i i'}{n_k}\frac{[(n_k-i)\theta^{n_k-i-1}-i\theta^{i-1}](1-\theta^{i'})+i'(\theta^{n_k-i}-\theta^i)\theta^{i'-1}}{(1-\theta^{i'})^2}. \tag{75}$$

Hence, for any $\theta_{\max}\in(0,1)$,

$$\frac{d\gamma(\theta)}{d\theta}\leq L:=\sum_{i'=1}^{n_k}\sum_{i=n_k^0}^{n_k}\frac{\alpha_i i'}{n_k}\frac{(n_k-i)\theta_{\max}^{n_k-i-1}+i'\theta_{\max}^{n_k-i}\theta_{\max}^{i'-1}}{(1-\theta_{\max}^{i'})^2} \tag{76}$$

for all $\theta\in(0,\theta_{\max}]$. Consequently, $\gamma(\cdot)$ is Lipschitz continuous on $(0,\theta_{\max}]$ with Lipschitz constant L. Hence, it follows that it suffices to check $dg(\theta_{N_k})/da$ for $n_k\in\{2,3,...,1407\}$ and a finite number of values for θ_{N_k} to verify that $dg(\theta_{N_k})/da<1$ for all $\theta_{N_k}\in(0,\theta_{\max}]$. Let $\tilde{\theta}_1$, $\tilde{\theta}_2$, ..., $\tilde{\theta}_{\tilde{k}}$ be these values, which are computed recursively starting with $\tilde{\theta}_1=0$ and then by $\tilde{\theta}_{k+1}=\tilde{\theta}_k+(1-\gamma(\tilde{\theta}_k))/L$, $k=1,2,...$, until a value no smaller than $\theta_{\max}$ is obtained. Let $\theta_{\max}=0.99$. Since we find that $\gamma(\tilde{\theta}_k)<1$ for all k in this case, it follows from the fact that $\gamma(\cdot)$ is Lipschitz continuous on

$(0, 0.99]$ with Lipschitz constant L that $\gamma(\theta) < 1$ for all $\theta \in [0, 0.99]$. Hence, $dg(\theta_{N_k})/da < 1$ for all $\theta_{N_k} \in (0, 0.99]$ and $n_k = 2, 3, ..., 1407$. The conclusion then follows by the fixed-point theorem. □

Proof of Proposition 4. By Proposition 1, $N^{1/2}(f_N^* - f^*) \Rightarrow \mathcal{N}(0, \sigma^2(x^*))$, as $N \to \infty$. Let $\{N_l(S_k)\}_{S_k=1}^{\infty}$, $l = 1, 2, ..., k$, be such that $N_l(S_k) \in \mathbb{N}$ for all $S_k \in \mathbb{N}$ and $l = 1, 2, ..., k$, $\sum_{l=1}^{k} N_l(S_k) = S_k$, and $N_l(S_k)/S_k \to \beta_l \in [0, 1]$, as $S_k \to \infty$. Consequently, $\sum_{l=1}^{k} \beta_l = 1$. By Slutsky's theorem (see, e.g., Exercise 25.7 of [7]), it then follows that for all $l = 1, 2, ..., k$,

$$\left(\frac{N_l(S_k)}{S_k}\right)^{1/2} N_l(S_k)^{1/2}(f_{N_l(S_k)}^* - f^*) \Rightarrow \beta_l^{1/2}\mathcal{N}(0, \sigma^2(x^*)), \tag{77}$$

as $S_k \to \infty$.

Since the sequences $\{f_{N_l}(x_i^l)\}_{i=0}^{n_l}$, $l = 1, 2, ..., k$, converge exactly linearly with coefficient $\hat{\theta}_{l+1}$, it follows that the minimization in (43) can be ignored and $\hat{m}_l = f_{N_l}^*$, $l = 1, 2, ..., k$. Using the recursive formula for $\hat{f}_{k+1}^*$ in Step 2 of Subroutine C, we find that $\hat{f}_{k+1}^* = \sum_{l=1}^{k} N_l(S_k) f_{N_l}^* / S_k$. Consequently,

$$S_k^{1/2}(\hat{f}_{k+1}^* - f^*) = \sum_{l=1}^{k} \left(\frac{N_l(S_k)}{S_k}\right)^{1/2} N_l(S_k)^{1/2}(f_{N_l(S_k)}^* - f^*). \tag{78}$$

It then follows by the continuous mapping theorem and the independence of samples across stages that

$$\sum_{l=1}^{k} \left(\frac{N_l(S_k)}{S_k}\right)^{1/2} N_l(S_k)^{1/2}(f_{N_l(S_k)}^* - f^*) \Rightarrow \sum_{l=1}^{k} \beta_l^{1/2}\mathcal{N}(0, \sigma^2(x^*)) = \mathcal{N}(0, \sum_{l=1}^{k} \beta_l \sigma^2(x^*)), \tag{79}$$

as $S_k \to \infty$. The conclusion then follows from the fact that $\sum_{l=1}^{k} \beta_l = 1$. □

References

[1] S. Alexander, T.F. Coleman, and Y. Li. Minimizing CVaR and VaR for a portfolio of derivatives. *J. Banking & Finance*, 30:583–605, 2006.

[2] H. Attouch and R. J-B Wets. Epigraphical processes: laws of large numbers for random lsc functions. In *Seminaire d'Analyze Convexe*, pages 13.1–13.29. Montpellier, 1990.

[3] F. Bastin, C. Cirillo, and P.L Toint. An adaptive monte carlo algorithm for computing mixed logit estimators. *Computational Management Science*, 3(1):55–79, 2006.

[4] G. Bayraksan and D.P. Morton. A sequential sampling procedure for stochastic programming. *Operations Research*, 59(4):898–913, 2011.

[5] D. P. Bertsekas. *Dynamic programming and optimal control*. Athena Scientific, Belmont, Massachusetts, 3. edition, 2007.

[6] J.T. Betts and W.P Huffman. Mesh refinement in direct transcription methods for optimal control. *Optimal Control Applications*, 19:1–21, 1998.

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

[8] G. Deng and M.C. Ferris. Variable-number sample-path optimization. *Mathematical Programming B*, 117:81–109, 2009.

[9] Y. Ermoliev. Stochastic quasigradient methods. In *Numerical Techniques for Stochastic Optimization*, Yu. Ermoliev and R.J-B. Wets (Eds.), New York, New York, 1988. Springer.

[10] P. E. Gill, S. J. Hammarling, W. Murray, M. A. Saunders, and M. H. Wright. LSSOL 1.0 User's guide. Technical Report SOL-86-1, System Optimization Laboratory, Stanford University, Stanford, California, 1986.

[11] M. Grant and S. Boyd. CVX: Matlab software for disciplined convex programming, version 1.21. http://cvxr.com/cvx, December 2010.

[12] L. He and E. Polak. Effective diagonalization strategies for the solution of a class of optimal design problems. *IEEE Transactions on Automatic Control*, 35(3):258–267, 1990.

[13] J. L. Higle and S. Sen. *Stochastic Decomposition: A Statistical Method for Large Scale Stochastic Linear Programming*. Springer, 1996.

[14] K. Holmstrom. Tomlab optimization. http://tomopt.com, 2009.

[15] T. Homem-de-Mello. Variable-sample methods for stochastic optimization. *ACM Transactions on Modeling and Computer Simulation*, 13(2):108–133, 2003.

[16] T. Homem-de-Mello, A. Shapiro, and M.L. Spearman. Finding optimal material release times using simulation-based optimization. *Management Science*, 45(1):86–102, 1999.

[17] J. Hu, M.C. Fu, and S.I. Marcus. A model reference adaptive search method for global optimization. *Operations Research*, 55(3):549–568, 2007.

[18] G. Infanger. *Planning under uncertainty: solving large-scale stochastic linear programs*. Thomson Learning, 1994.

[19] P. Kall and J. Meyer. *Stochastic Linear Programming, Models, Theory, and Computation*. Springer, 2005.

[20] W. Kohn, Z.B. Zabinsky, and V. Brayman. Optimization of algorithmic parameters using a meta-control approach. *J. Global Optimization*, 34:293–316, 2006.

[21] H. J. Kushner and G. G. Yin. *Stochastic Approximation and Recursive Algorithms and Applications*. Springer, 2. edition, 2003.

[22] G. Lan. *Convex Optimization Under Inexact First-order Information*. Phd thesis, Georgia Institute of Technology, Atlanta, Georgia, 2009.

[23] J. Linderoth, A. Shapiro, and S. Wright. The empirical behavior of sampling methods for stochastic programming. *Annals of Operations Research*, 142:215–241, 2006.

[24] W. K. Mak, D. P. Morton, and R. K. Wood. Monte Carlo bounding techniques for determining solution quality in stochastic programs. *Operations Research Letters*, 24:47–56, 1999.

[25] O. Molvalioglu, Z.B. Zabinsky, and W. Kohn. The interacting-particle algorithm with dynamic heating and cooling. *J. Global Optimization*, 43:329–356, 2009.

[26] T. Munakata and Y. Nakamura. Temperature control for simulated annealing. *Physical Review E*, 64(4):46–127, 2001.

[27] A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM J. Optimization*, 19(4):1574–1609, 2009.

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

[29] J. Oppen and D.L. Woodruff. Parametric models of local search progress. *International Transactions in Operational Research*, 16:627–640, 2009.

[30] R. Pasupathy. On choosing parameters in retrospective-approximation algorithms for stochastic root finding and simulation optimization. *Operations Research*, 58:889–901, 2010.

[31] E.Y. Pee and J. O. Royset. On solving large-scale finite minimax problems using exponential smoothing. *Journal of Optimization Theory and Applications*, 148(2):390–421, 2011.

[32] O. Pironneau and E. Polak. Consistent approximations and approximate functions and gradients in optimal control. *SIAM J. Control and Optimization*, 41(2):487–510, 2002.

[33] E. Polak. *Optimization. Algorithms and consistent approximations*. Springer, New York, New York, 1997.

[34] E. Polak and J. O. Royset. Efficient sample sizes in stochastic nonlinear programming. *J. Computational and Applied Mathematics*, 217:301–310, 2008.

[35] E. Polak, J. O. Royset, and R. S. Womersley. Algorithms with adaptive smoothing for finite minimax problems. *J. Optimization. Theory and Applications*, 119(3):459–484, 2003.

[36] R.T. Rockafellar and S. Uryasev. Conditional value-at-risk for general loss distributions. *Journal of Banking and Finance*, 26:1443–1471, 2002.

[37] J. O. Royset. Optimality functions in stochastic programming. *Mathematical Programming*, to appear, 2012.

[38] J. O. Royset and E. Polak. Implementable algorithm for stochastic programs using sample average approximations. *J. Optimization. Theory and Application*, 122(1):157–184, 2004.

[39] J. O. Royset and E. Polak. Extensions of stochastic optimization results from problems with simple to problems with complex failure probability functions. *J. Optimization. Theory and Application*, 133(1):1–18, 2007.

[40] J. O. Royset, E. Polak, and A. Der Kiureghian. Adaptive approximations and exact penalization for the solution of generalized semi-infinite min-max problems. *SIAM J. Optimization*, 14(1):1–34, 2003.

[41] J.O. Royset and E. Polak. Sample average approximations in reliability-based structural optimization: Theory and applications. In M. Papadrakakis Y. Tsompanakis, N.D. Lagaros, editor, *Structural design optimization considering uncertainties*, pages 307–334. Taylor & Francis, 2008.

[42] R.Y. Rubinstein and D.P. Kroese. *The cross-entropy method: a unified combinatorial approach to combinatorial optimization, Monte-Carlo simulation, and machine learning*. Springer, 2004.

[43] K. Sastry and D.E. Goldberg. Let's get ready to rumble redux: crossover versus mutation head to head on exponentially scaled problems. In *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 1380–1387, New York, NY, USA, 2007. ACM.

[44] A. Schwartz and E. Polak. Consistent approximations for optimal control problems based on Runge-Kutta integration. *SIAM J. Control and Optimization*, 34(4):1235–1269, 1996.

[45] A. Shapiro. Asymptotic analysis of stochastic programs. *Annals of Operations Research*, 30:169–186, 1991.

[46] A. Shapiro, D. Dentcheva, and A. Ruszczynski. *Lectures on Stochastic Programming: Modeling and Theory*. Society of Industrial and Applied Mathematics, 2009.

[47] A. Shapiro and T. Homem-de-Mello. A simulation-based approach to two-stage stochastic programming with recourse. *Mathematical Programming*, 81:301–325, 1998.

[48] A. Shapiro and Y. Wardi. Convergence analysis of stochastic algorithms. *Mathematics of Operations Research*, 21(3):615–628, 1996.

[49] J. C. Spall. *Introduction to stochastic search and optimization*. John Wiley and Sons, New York, New York, 2003.

[50] B. Verweij, S. Ahmed, A. J. Kleywegt, G. Nemhauser, and A. Shapiro. Sample average approximation method applied to stochastic routing problems: a computational study. *Computational Optimization and Applications*, 24(2-3):289–333, 2003.

[51] A. R. Washburn. *Search and Detection*. INFORMS, Linthicum, Maryland, 4. edition, 2002.

[52] H. Xu and D. Zhang. Smooth sample average approximation of stationary points in nonsmooth stochastic optimization and applications. *Mathematical Programming*, 119:371–401, 2009.

[53] S. Xu. Smoothing method for minimax problems. *Computational Optimization and Applications*, 20:267–279, 2001.