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

December 21, 2009

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 sample average approximations (SAA). 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 SAA 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 approximations (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. SAA is intuitive, simple, and has a strong theoretical foundation; see Chapter 5 of [41] for a summary of results and [20, 45, 19, 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

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
21 DEC 2009		**00-00-2009 to 00-00-2009**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
On Sample Size Control in Sample Average Approximations for Solving Smooth Stochastic Programs	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Naval Postgraduate School,Operations Research Department,Monterey,CA,93943**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
in review

14. ABSTRACT
see report

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**28**	

reduce the computing time compared to a fixed sample-size policy [42, 13, 12, 2, 34, 29, 3, 26]. 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., [43]. There is also ample empirical evidence from other fields such as semi-infinite programming [9, 35], minimax optimization [48, 30], and optimal control [39, 4, 27] 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 SAA 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, this paper deals with the specific case of smooth stochastic programs where the sample average problems are approximately solved by standard nonlinear programming algorithms. Consequently, we assume that the sample average problems are smooth and their gradients can be computed relatively easily. This case arises for example in estimation of mixed logit models [2], search theory (see Section 5), and engineering design [33]. Important models such as two-stage stochastic programs with recourse [16], conditional Value-at-Risk minimization [31], inventory control problems [47], and complex engineering design problems [34] involve nonsmooth random functions and sample average problems. However, recent efforts to apply smooth approximations of nonsmooth random functions appear promising [1, 47]. Hence, the results of this paper may also be applicable in such contexts. We illustrate the use of smooth approximations in Section 5. 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 [44, 37, 14, 6] for an overview of that area of research. We note that stochastic programs may also be solved by stochastic approximations [7, 18, 23] and stochastic decomposition [10, 15] under suitable assumptions. However, in this paper we focus on SAA.

Existing sample-size selection policies for SAA 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 [43, 42, 13, 2, 34, 26]. We also find studies of consistency of SAA estimators defined by variable sample sizes [12].

The issue of determining a computationally efficient sample-size selection policies has received much less attention than that of asymptotic convergence. The recent paper [26] 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 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 [26] 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 [29], 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 or other algorithm parameters in the areas of semi-infinite programming [9], interior-point methods [17], interacting-particle algorithms [21], and simulated annealing [22].

While we here focus on obtaining a near-optimal solution, [3] deals with how to efficiently estimate the quality of a given sequence of candidate solutions. The paper provides rules for selecting variable sample sizes for the 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 SAA 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 [17] in the context of interior-point methods for solving deterministic nonlinear programs and in [21] for interacting-particle algorithms. Since SAA 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 [25, 38]. The surrogate model leads to a surrogate discrete-time optimal-control problem that we solve approximately by dynamic programming.

While the auxiliary nonlinear program for sample-size selection in [29] 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 SAA and the possibility of recourse in future stages of the calculations. As the surrogate model depends on unknown parameters, we approximately solve the optimal-control problem 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 [26] and other plausible polices.

Our sample-size selection policy does not include hard-to-select user specified parameters that may greatly influence computing times. Hence, the 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 as a discrete-time optimal-control problem. The optimal-control problem appears to be unsolvable 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* $F : \mathbb{R}^d \times \Omega \to \mathbb{R}$. 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 $|F(x, \omega)|$ is bounded by an integrable function 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 [41]). 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 general probability distributions $\mathbb{P}$, 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* $A_N : X \to X$. While we state the sample-size control problem below without further assumptions, we essentially throughout this paper assume that the algorithm map is linearly convergent as formally stated in Section 3. There may be many algorithms that obtain linear convergence for a given $f_N(\cdot)$. We are motivated by situations where $F(\cdot, \omega)$ is continuously differentiable for $\mathbb{P}$-almost every $\omega \in \Omega$. In such situations the algorithm map may be defined by one iteration of the projected gradient method (see, e.g., p. 66 of [28]) or some other nonlinear programming algorithm applied to $\mathbf{P}_N$. We note that [26] also considers linearly convergent algorithm maps.

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 $F(\cdot, \cdot)$ and independent sampling, $f_N(x)$ converges to $f(x)$ uniformly on X, as $N \to \infty$, almost surely; see for example Theorem 7.48 in [41]. Now suppose that we apply an algorithm map $A_N(\cdot)$ to $f_N(\cdot)$ that is convergent in the following sense: for all N and any sequence $\{x_i\}_{i=0}^{\infty}$ generated by the iteration $x_{i+1} = A_N(x_i)$, the sequence $\{x_i\}_{i=0}^{\infty}$ tends to a point in $\hat{X}^*_N$ almost surely. Then, for any $\epsilon > 0$, a sufficiently large N and a

sufficiently large number of iterations of the algorithm map result in 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 when starting sufficiently close to $\hat{X}_N^*$. 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 [12]. After the sample generation, n_k iterations with the algorithm map $A_{N_k}(\cdot)$ are carried out on $\mathbf{P}_{N_k}$ using the generated sample. This approach is described next in a conceptual algorithm.

Algorithm 1 (Conceptual Algorithm for P).

Data. Optimality tolerance $\epsilon > 0$; initial solution $x_0^1 \in X$.

Step 0. Set stage counter $k = 1$.

Step 1. Determine number of iterations $n_k \in \mathbb{N}$ and sample size $N_k \in \mathbb{N}$.
Generate an independent sample of size N_k.

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 1.

Step 3. If $x_{n_k}^k \in X_\epsilon^*$, then **Stop**. Else, set $x_0^{k+1} = x_{n_k}^k$, replace k by $k+1$, and go to **Step 1**.

In view of the discussion above, it is clear that given a particular stage k and a convergent algorithm map, there exists an N' and n' such that if $N_k \geq N'$ and $n_k \geq n'$, then Algorithm 1 stops after the k-th stage almost surely.

We note that Algorithm 1 resembles the classical batching approach to obtain a lower bound on the optimal value of a two-stage stochastic program with recourse [20]. In that case, M 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 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.

Clearly, Step 3 of Algorithm 1 is not implementable as X_ϵ^* is defined in terms of the unknown f^* and the uncomputable $f(\cdot)$. We discuss this further in Section 4. Given $\mathbf{P}$ and the families $\{\mathbf{P}_N\}_{N\in\mathbb{N}}$ and $\{A_N(\cdot)\}_{N\in\mathbb{N}}$, our goal is to determine an efficient policy for selecting the number of iterations and the sample size in Step 1 of Algorithm 1. Specifically, we would like to find a policy 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.

2.2 Sample-Size Control Problem

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 $i = 0, 1, 2, ..., n-1$,

$$A_N^{i+1}(x) = A_N(A_N^i(x)). \tag{6}$$

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

$$x_{n_k}^k = A_{N_k}^{n_k}(x_{n_{k-1}}^{k-1}), k = 1, 2, 3, ..., \tag{7}$$

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 $x_{n_0}^0 = x_0^1$ is the *initial condition*. We denote the random sample of stage k by $\bar{\omega}^k = (\omega_1^k, \omega_2^k, ..., \omega_{N_k}^k)$. Clearly, for any $k \in \mathbb{N}$, $x_{n_k}^k$ is unknown prior to the realization of the samples $\bar{\omega}^1$, $\bar{\omega}^2$, ..., $\bar{\omega}^k$. Hence, $\bar{\omega}^k$ is the *disturbance* induced at the k-th stage.

We define the feasible set of controls $U(x)$ as follows: If $x \in X_\epsilon^*$, then $U(x) = \{(1, 0)\}$. Otherwise, $U(x) = \mathbb{N} \times \mathbb{N}$. 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 $A_N(\cdot)$. Also, we set $c(1, 0) = 0$.

Given an initial solution $x_0^1 \in X$, we seek an admissible stationary policy $\mu : X \to \mathbb{N} \times (\mathbb{N} \cup \{0\})$ with $\mu(x_{n_{k-1}}^{k-1}) \in U(x_{n_{k-1}}^{k-1})$ for all $x_{n_{k-1}}^{k-1} \in X$, $k \in \mathbb{N}$, that minimizes the *total cost function*

$$J_\mu(x_0^1) := \limsup_{s\to\infty} E\left[\sum_{k=1}^{s} c(\mu(x_{n_{k-1}}^{k-1}))\right] \tag{8}$$

subject to the constraints (7). (In (8) we slightly abuse notation by allowing $c(\cdot, \cdot)$ to take a two-dimensional vector as input instead of two scalar values.) Here, E denotes expectation with respect to the disturbances due to the samples $\bar{\omega}^1$, $\bar{\omega}^2$, ..., $\bar{\omega}^s$. We assume that the cost function $c(\cdot, \cdot)$, policy $\mu(\cdot)$, and algorithm map $A_N(\cdot)$ satisfy sufficient measurability assumptions so that this expectation is defined.

For a given initial solution $x_0^1 \in X$, we define the *sample-size control problem*

$$\textbf{SSCP}: \quad \inf_{\mu} J_\mu(x_0^1), \tag{9}$$

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

Clearly, 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 (7) can only be evaluated by computational costly calculations, and (iv) the expectation in (8) 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 parts, overcome these difficulties.

3 Surrogate Sample-Size Control Problem

Instead of attempting to solve **SSCP**, we construct and approximately solve a surrogate sample-size control problem. 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

We assume that the algorithm map $A_N(\cdot)$ used in Algorithm 1 is uniformly linearly convergent as made precise in the following assumption.

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

$$f_N(A_N(x)) - f_N^* \leq \theta(f_N(x) - f_N^*) \quad \text{a.s.} \tag{10}$$

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, see, e.g., Theorem 1.3.18 in [28]. Assumption 1 requires that there exists a uniform rate of convergence coefficient that is valid almost surely. This holds, for instance, when the eigenvalues of the Hessian of $f_N(x)$, $x \in X$, $N \in \mathbb{N}$, are positive and are bounded from above and away from zero almost surely. We find a similar linear rate of convergence assumption in [26].

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

$$f_N^* \leq f_N(A_N^n(x)) \leq \bar{f}_N^n(x) := f_N^* + \theta^n(f_N(x) - f_N^*) \quad \text{a.s.} \tag{11}$$

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 (11). The derivations rely on the following assumptions.

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

Assumption 3 *There exists a measurable function* $C : \Omega \to [0,\infty)$ *such that* $\mathbb{E}[C(\omega)^2] < \infty$ *and*

$$|F(x,\omega) - F(x',\omega)| \leq C(\omega)\|x - x'\| \tag{12}$$

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 (11) is typically "near" f^* for large N as stated next.

Proposition 1 *[40] Suppose that Assumptions 2 and 3 hold. Then,*

$$N^{1/2}(f_N^* - f^*) \Rightarrow \inf_{x \in X^*} Y(x), \tag{13}$$

as $N \to \infty$.

Consequently, if there is a unique optimal solution x^* of $\mathbf{P}$, i.e., $X^* = \{x^*\}$, then the lower bound f_N^* on $f_N(A_N^n(x))$ (see (11)) 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 1 *Suppose that Assumptions 1, 2, and 3 hold and that there is a unique optimal solution* x^* *of* $\mathbf{P}$*, i.e.,* $X^* = \{x^*\}$. *Then, for any* $x \in X$ *and* $n \in \mathbb{N}$

$$N^{1/2}[\bar{f}_N^n(x) - f^* - \theta^n(f(x) - f^*)] \Rightarrow \mathcal{N}(0, v_n(x)), \tag{14}$$

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

$$v_n(x) = (1-\theta^n)^2\sigma^2(x^*) + \theta^{2n}\sigma^2(x) + 2Cov(F(x^*,\omega), F(x,\omega))(1-\theta^n)\theta^n. \tag{15}$$

Proof: By definition

$$\begin{aligned} N^{1/2}[\bar{f}_N^n(x) - f^* - \theta^n(f(x) - f^*)] &= (1-\theta^n)N^{1/2}(f_N^* - f^*) \\ &+ \theta^n N^{1/2}(f_N(x) - f(x)). \end{aligned} \tag{16}$$

Since **P** has a unique optimal solution, it follows from Theorem 5.7 in [41] that

$$N^{1/2}\begin{pmatrix} f_N(x) - f(x) \\ f_N^* - f^* \end{pmatrix} \Rightarrow \begin{pmatrix} Y(x) \\ Y(x^*) \end{pmatrix}, \tag{17}$$

as $N \to \infty$. Then, the result follows after application of the continuous mapping theorem, see, e.g., Theorem 29.2 in [5]. □

In view of Theorem 1, 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 **P**, we obtain the following asymptotic results as $n, N \to \infty$.

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

$$\theta^{-n}[\bar{f}_N^n(x) - f^*] \Rightarrow f(x) - f^*, \text{ if } a = \infty; \tag{18}$$

$$N^{1/2}[\bar{f}_N^n(x) - f^*] \Rightarrow \inf_{x' \in X^*} Y(x') + a(f(x) - f^*), \text{ if } a \in [0, \infty); \tag{19}$$

as $N, n \to \infty$.

Proof: We only consider (19) as the other case follows by similar arguments. By definition,

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

The result now follows from Proposition 1, the central limit theorem, and Slutsky's theorem (see, e.g., Exercise 25.7 of [5]). □

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

$$N^{1/2}[f_N(A_N^n(x)) - f^*] \Rightarrow \inf_{x' \in X^*} Y(x') \tag{21}$$

as $N, n \to \infty$.

Proof: The result follows directly from (11), Proposition 1, and Theorem 2. □

In view of Theorem 2, 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 observations is approximately equivalent to the one after Theorem 1 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 1 and 2 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 [25] and genetic algorithms [38]. However, these models appear not directly 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 dimensions to one dimension. Theorems 1 and 2 indicate possible models for the evolution of function values in Algorithm 1. If n_k and N_k are large, Theorem 2 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

$$\mathcal{N}(f^* + \theta^{n_k}(f(x_{n_{k-1}}^{k-1}) - f^*), \sigma^2(x^*)/N_k). \tag{22}$$

Moreover, if only N_k is large, Theorem 1 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

$$\mathcal{N}(f^* + \theta^{n_k}(f(x_{n_{k-1}}^{k-1}) - f^*), v_{n_k}/N_k). \tag{23}$$

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, (22) and (23) are approximately equal in distribution when $x_{n_{k-1}}^{k-1}$ is close to x^*. The paragraph after Corollary 1 indicates that (22) and (23) are also approximately equal in distribution when n_k is large. Consequently, we adopt the simpler expression (22) 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. (This conjecture is supported by our numerical experiments.) On the other hand, when k is large, $x_{n_{k-1}}^{k-1}$ tends to be close to x^*. Hence, (22) appears to be reasonably accurate in the present context. We have verified empirically that (23) is not a significantly better approximation of $f_{N_k}(x_{n_k}^k)$ than (22).

Clearly, (22) would not be sufficient for estimating $f(x_{n_k}^k)$, $f(x_{n_{k+1}}^{k+1})$, etc., as knowledge of $f(x_{n_{l-1}}^{l-1})$ does not specify $f(x_{n_l}^l)$, only $f_{N_l}(x_{n_l}^l)$. We are unable to derive the distribution of $f(x_{n_k}^k)$ conditional on $x_{n_{k-1}}^{k-1}$ and heuristically approximate that distribution by (22) with truncation at f^* to account for the fundamental relation $f(x) \geq f^*$ for all $x \in X$. Hence, we let

$$\bar{\mathcal{N}}(f^* + \theta^{n_k}(f(x_{n_{k-1}}^{k-1}) - f^*), \sigma^2(x^*)/N_k, f^*) \tag{24}$$

be our approximation of the distribution of $f(x_{n_k}^k)$ conditional on $x_{n_{k-1}}^{k-1}$, where $\bar{\mathcal{N}}(m, v, t)$ denotes a truncated normally distributed random variable with an underlying normal distribution $\mathcal{N}(m, v)$ and lower truncation thresholds t, i.e., the cumulative distribution function $\bar{\Phi}(\xi)$ of $\bar{\mathcal{N}}(m, v, t)$ is

$\bar{\Phi}(\xi) = (\Phi((\xi - m)/\sqrt{v}) - \Phi((t - m)/\sqrt{v}))/(1 - \Phi((t - m)/\sqrt{v}))$, $\xi \geq t$, 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 (24) to estimate $f(x_{n_k}^k)$. Moreover, we could use (24) 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 at that time. Hence, we proceed with (24) 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.

Given the estimates $p_f, p^*, p_\theta, p_\sigma$ and the controls (N_k, n_k), (N_{k+1}, n_{k+1}), $(N_{k+2}, n_{k+2}), ...,$ we define the *surrogate dynamic equation* for the surrogate state by

$$f_{l+1} = \bar{\mathcal{N}}(p^* + p_\theta^{n_l}(f_l - p^*), p_\sigma^2/N_l, p^*), \quad l = k, k+1, k+2, ..., \tag{25}$$

with initial condition $f_k = p_f$. We note that the equality in (25) indicates equality in distribution. The terminal states of **SSCP** are defined by X_ϵ^*, which translates to the following *surrogate terminal states*

$$T := \{\xi \in \mathbb{R} | \xi - p^* \leq \epsilon\}. \tag{26}$$

We define the feasible set of controls $R(\xi)$ for the surrogate problem as follows: If $\xi \in T$, then $R(\xi) := \{(1, 0)\}$. Otherwise, $R(\xi) := \mathbb{N} \times \mathbb{N}$.

We now define the surrogate sample-size control problem. Given a stopping tolerance $\epsilon > 0$ and the estimates p_f, p^*, p_θ, and p_σ at the beginning of stage k, we seek an admissible stationary policy $\mu : \mathbb{R} \to \mathbb{N} \times (\mathbb{N} \cup \{0\})$ with $\mu(f_l) \in R(f_l)$ for all $f_l \in \mathbb{R}$, $l = k, k+1, k+2, ...,$ that minimizes the *total surrogate cost function*

$$J_{k,\mu}(p_f, p^*, p_\theta, p_\sigma, \epsilon) := \limsup_{s \to \infty} E\left[\sum_{l=k}^{k+s} c(\mu(f_l))\right] \tag{27}$$

subject to the constraints (25). Here, E denotes expectation with respect to the disturbances on stages k, $k+1$, ..., $k+s$ given by the truncated normal distribution in (25). Then, we define the *surrogate sample-size control problem*

$$\textbf{S-SSCP}_k(p_f, p^*, p_\theta, p_\sigma, \epsilon): \quad \inf_\mu J_{k,\mu}(p_f, p^*, p_\theta, p_\sigma, \epsilon), \tag{28}$$

where the infimum is over all admissible policies.

We could easily restrict $R(\xi)$ in **S-SSCP**$_k$. For example, one could impose minimum values on the sample size and the number of iterations at various stages to ensure that the surrogate dynamic equation (25) represents the algorithmic progress reasonably accurately. In numerical examples, we effectively impose such restrictions as part of the solution process as described next.

We solve **S-SSCP**$_k$ approximately by discretizing the surrogate state space and the truncated normal distribution, and then apply the backward recursion algorithm to the resulting dynamic program. In the numerical examples below, we find it sufficient to consider 10 stages into the future and to discretize the interval $[p^* + \epsilon, p_f + 1.96 p_\sigma / N_{k-1})$ using 14 points. We only consider 10 possible values of n_l in the range zero to $\max\{10, \lceil \log(0.1\epsilon/(p_f - p^*))/\log p_\theta \rceil\}$, where $\lceil a \rceil$ denotes the smallest integer no smaller than a. We observe that the upper end of the range of n_l is simply the larger of 10 and the number of iterations required to reach within 0.1ϵ of the optimal value in presence of no uncertainty. We consider 18 possible values of N_l mostly in the range $\lceil 1.1 N_{k-1} \rceil$ and $10 N_{k-1}$, but also with two larger values dynamically selected.

The optimal policy found from solving the discretized **S-SSCP**$_k$ provides controls (N_k, n_k), (N_{k+1}, n_{k+1}), (N_{k+2}, n_{k+2}), etc. 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 after each stage. We refer to the resulting policy as the S-SSCP policy. We discuss the parameter estimation and the full algorithm next.

4 Parameter Estimation and Full Algorithm

4.1 Parameter Estimation

After completing n_k iterations with sample size N_k in stage k of Algorithm 1, 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 use these quantities to update the parameters p_f, p^*, p_θ, and p_σ.

We estimate $\sigma^2(x^*)$ for stage $k+1$ by simply setting it equal to the sample variance at the last iterate, i.e.,

$$p_\sigma^2 = \hat{\sigma}_{k+1}^2 := \frac{1}{N_k - 1} \sum_{j=1}^{N_k} (F(x_{n_k}^k, \omega_j) - f_{N_k}(x_{n_k}^k))^2, \tag{29}$$

where $\omega_1, \omega_2, ..., \omega_{N_k}$ is the sample used in stage k.

We adopt the approach in [9] to estimate the rate of convergence coefficient θ, but add an exponential smoothing step to avoid large changes in the estimate. A (biased) estimate of f^* is computed by a weighted average of estimates of $f_{N_l}^*$. We let $\hat{f}_k^*$ and $\hat{\theta}_k$ denote the estimates of f^* and θ at the beginning of the k-th stage, respectively.

Subroutine A (Estimates θ and f^* at the end of stage k).

Parameters. Exponential smoothing parameter $\phi \in (0, 1]$ and tolerance $\epsilon_\theta > 0$.

Data. Previous estimates $\hat{\theta}_k$ and $\hat{f}_k^*$; and function values $\{f_{N_k}(x_i^k)\}_{i=0}^{n_k}$ of stage k.

Step 0. Set $\hat{\theta} = \hat{\theta}_k$.

Step 1. Estimate minimum value of $\mathbf{P}_{N_k}$:

$$d_k = \frac{1}{n_k} \sum_{i=0}^{n_k-1} \frac{f_{N_k}(x_{n_k}^k) - \hat{\theta}^{n_k-i} f_{N_k}(x_i^k)}{1 - \hat{\theta}^{n_k-i}}. \tag{30}$$

Step 2. Solve the least-square problem

$$(a^*, b^*) = \arg\min_{a,b} \sum_{i=0}^{n_k} (\log(f_{N_k}(x_i^k) - d_k) - i \log a - b)^2. \tag{31}$$

Step 3. If $|\hat{\theta} - a^*| < \epsilon_\theta$, set $\hat{\theta} = a^*$ and go to **Step 4**. Else, set $\hat{\theta} = a^*$ and go to **Step 1**.

Step 4. Set $\hat{\theta}_{k+1} = \phi\hat{\theta} + (1-\phi)\hat{\theta}_k$.

Step 5. Compute conservative estimate of minimum value of $\mathbf{P}_{N_k}$:

$$\hat{m}_k := \min_{i=0,1,\ldots,n_k-1} \frac{f_{N_k}(x_{n_k}^k) - (\hat{\theta}_{k+1})^{n_k-i} f_{N_k}(x_i^k)}{1 - (\hat{\theta}_{k+1})^{n_k-i}}. \tag{32}$$

Step 6. Estimate optimal value of $\mathbf{P}$:

$$\hat{f}_{k+1}^* := \frac{N_k}{\sum_{l=1}^k N_l} \hat{m}_k + \frac{\sum_{l=1}^{k-1} N_l}{\sum_{l=1}^k N_l} \hat{f}_k^* \tag{33}$$

and **Stop**.

It follows from Assumption 1 that $f_{N_k}^* \geq (f_{N_k}(x_{n_k}^k) - \theta^{n_k-i} f_{N_k}(x_i^k))/(1-\theta^{n_k-i})$ almost surely for any $i = 0, 1, ..., n_k - 1$. Step 1 in Subroutine A averages these lower bounding estimates with the current estimate of θ and uses that as an estimate of $f_{N_k}^*$. Given the estimate of $f_{N_k}^*$, Step 2 computes the rate of convergence coefficient that best fit $\{f_{N_k}(x_i^k)\}_{i=0}^{n_k}$ in a least-square sense. We observe that if $\{f_{N_k}(x_i^k)\}_{i=0}^{n_k}$ is a linearly progressing sequence, then a correct θ is found by Steps 1-3 of Subroutine A. Those steps were originally proposed in [9] in the context of semi-infinite optimization.

Step 4 applies exponential smoothing to the estimates of the rate of convergence coefficient to avoid large fluctuations. Step 5 computes a conservative estimate of $f_{N_k}^*$ that is merged with the current estimate of f^* using weighted averages in Step 6.

At the end of the k-th stage, we set $p_\theta = \hat{\theta}_{k+1}$, i.e., the estimate of the rate of convergence coefficient using iterates up to stage k. Typically, we set $p^* = \hat{f}_{k+1}^*$, but adjust that if a condition is satisfied as described in the next subsection.

For all $k \in \mathbb{N}$, we estimate $f(x_{n_k}^k)$ at the end of the k-th stage by computing the unbiased estimate $f_{N^*}(x_{n_k}^k)$ using an independent sample of size N^*. We typically set $p_f = f_{N^*}(x_{n_k}^k)$ for use in **S-SSCP**$_{k+1}$. However, we deviate from that occasionally as described below.

In **S-SSCP**$_k$, the computational cost function $c(\cdot,\cdot)$ is assumed to be known. In reality, that may not be the case and we adopt the following simple model of computational cost for one stage given N_k, n_k, and N^*. We set

$$c(N_k, n_k) = w_1 N_k n_k + w_2 N^* \tag{34}$$

where $w_1, w_2 \geq 0$ are parameters updated after the $(k-1)$-th stage by setting $w_1 = t_1/(N_{k-1}n_{k-1})$ and $w_2 = t_2/N^*$, with t_1 and t_2 being the computing times to carry out the n_{k-1} iterations and to compute the estimate $f_{N^*}(x_{n_{k-1}}^{k-1})$ during stage $k-1$, respectively. An alternative polynomial model of computational cost based on linear regression is used in [9]. However, we find (34) reasonable in the present situation as the time required to calculate $f_N(x)$ for a given x is linear in N. Hence, the effort required to apply the algorithm map once tends to be linear in N.

4.2 Full Algorithm

We now present the extension of Algorithm 1 that includes details about policy selection and parameter estimation.

Algorithm 2 (Implementable Algorithm for P).

Data. Optimality tolerance $\epsilon > 0$; initial sample size $N_0 \in \mathbb{N}$; validation sample size N^*; default sample size factor $\gamma_N > 0$; default iteration number $\gamma_n \in \mathbb{N}$; initial estimate of rate of convergence coefficient $\hat{\theta}_1$; initial solution $x_0^0 \in X$.

Step 0. Generate an independent sample of size N_0, compute $f_{N_0}(x_0^0)$ and $\hat{\sigma}_1$, and set $f_1 = \hat{f}_1^* = f_{N_0}(x_0^0)$, $p_f = f_1 + \hat{\sigma}_1/\sqrt{N_0}$, $p^* = \hat{f}_1^* - \max\{1, f_1\}$, $p_\theta = \hat{\theta}_1$, $p_\sigma = \hat{\sigma}_1$, $k = 1$, and $x_0^1 = x_0^0$.

Step 1. Determine n_k and N_k by solving **S-SSCP**$_k(p_f, p^*, p_\theta, p_\sigma, \epsilon)$.

Step 2. Generate an independent sample of size N_k.

Step 3. For $i = 0$ to $n_k - 1$: Compute $x_{i+1}^k = A_{N_k}(x_i^k)$ using the sample from Step 2.

Step 4. Compute $\hat{\sigma}_{k+1}$, $\hat{\theta}_{k+1}$, and $\hat{f}_{k+1}^*$ by (29) and Subroutine A.

Step 5. Generate an independent sample of size N^* and compute $f_{N^*}(x_{n_k}^k)$.

Step 6. **If** $\hat{f}_{k+1}^* + \epsilon < f_{N^*}(x_{n_k}^k)$, set $p_f = f_{N^*}(x_{n_k}^k)$, $p^* = \hat{f}_{k+1}^*$, $p_\theta = \hat{\theta}_{k+1}$, and $p_\sigma = \hat{\sigma}_{k+1}$. Replace k by $k+1$ and go to **Step 1**.

Elseif $\hat{f}_{k+1}^* - \hat{\sigma}_{k+1}/\sqrt{\sum_{l=1}^k N_l} + \epsilon < f_{N^*}(x_{n_k}^k) + \hat{\sigma}_{k+1}/\sqrt{N^*}$, then set $p_f = f_{N^*}(x_{n_k}^k) + \hat{\sigma}_{k+1}/\sqrt{N^*}$, $p^* = \hat{f}_{k+1}^* - \hat{\sigma}_{k+1}/\sqrt{\sum_{l=1}^k N_l}$, $p_\theta = \hat{\theta}_{k+1}$, and $p_\sigma = \hat{\sigma}_{k+1}$. Replace k by $k+1$ and go to **Step 1**.

Else set $N_{k+1} = \lceil \gamma_N N_k \rceil$ and $n_{k+1} = \gamma_n$. Replace k by $k+1$ and go to **Step 2**.

Step 0 of Algorithm 2 computes a rudimentary estimate of f^*. If problem specific information is available, the initial estimate of f^* may be improved. We note that if $\hat{f}^*_{k+1} + \epsilon \geq f_{N^*}(x^k_{n_k})$, then **S-SSCP**$_{k+1}(f_{N^*}(x^k_{n_k}), \hat{f}^*_{k+1}, p_\theta, p_\sigma, \epsilon)$ returns the policy $N_{k+1} = 1$ and $n_{k+1} = 0$ since a surrogate terminal state is already reached. This may occur if $\hat{f}^*_{k+1}$ and/or $f_{N^*}(x^k_{n_k})$ are inaccurately estimates of f^* and $f(x^k_{n_k})$, respectively. In that case, Step 6 of Algorithm 2 attempts to use conservative estimates of f^* and $f(x^k_{n_k})$. If the conservative estimates do not immediately lead to a surrogate terminal state, they are adopted in **S-SSCP**$_{k+1}(p_f, p^*, p_\theta, p_\sigma, \epsilon)$. If the conservative estimates still immediately lead to a surrogate terminal state, $x^k_{n_k}$ is probably (close to) a near-optimal solution and Algorithm 2 resorts to a default policy.

Even though **S-SSCP**$_k$ aims to obtain a near-optimal solution, such a solution is not guaranteed due to inaccurate parameter estimates and other approximations. However, Algorithm 2 is essentially identical to Algorithm 1 except it includes a specific policy for selecting N_k and n_k. Hence, under Assumption 1 and the adaption of a uniform law of large numbers (Theorem 7.48 in [41]), it follows that given a particular stage k, there exists an N' and n' such that if $N_k \geq N'$ and $n_k \geq n'$, then $x^k_{n_k}$ generated by Algorithm 2 is contained in X^*_ϵ. This fact gives assurance that a near-optimal solution can be obtained for large n_k and N_k.

In practice, all calculations must be terminated after a finite time period, which raises the question of how to estimate the proximity to optimality for the obtained solutions. We discuss that topic next.

4.3 Validation Analysis

The proximity to optimality of a solution $x^k_{n_k}$ obtained by Algorithm 2 could be estimated using an optimality function [32] or a hypothesis test of Karush-Kuhn-Tucker conditions [42]. We develop a third approach motivated by the optimality gap estimates of [24, 20]. As we see below, this approach utilizes quantities already computed in Algorithm 2.

Step 5 of Algorithm 2 computes in the k-th stage $f_{N^*}(x^k_{n_k})$. Under Assumption 2, the central limit theorem gives that $f_{N^*}(x^k_{n_k})$ is approximately normally distributed with mean $f(x^k_{n_k})$ and variance $\sigma^2(x^k_{n_k})/N^*$ for large N^*. Hence, $f_{N^*}(x^k_{n_k})$ is a probabilistic upper bound on f^*. This upper bound is identical to those used in [24, 20].

We depart from [24, 20] when constructing a probabilistic lower bound on f^*. If Assumption 1 holds and Subroutine A has constructed $\{\hat{\theta}_l\}_{l=2}^{k+1}$ and $\{\hat{m}_l\}_{l=1}^{k}$ such that $1 > \hat{\theta}_l \geq \theta$ for all $l = 2, 3, ..., k+1$, then $\hat{m}_l \leq f^*_{N_l}$ almost surely for all $l = 1, 2, 3, ..., k$. Consequently, it follows from the definition of $\hat{f}^*_{k+1}$ that

$$\hat{f}^*_{k+1} = \frac{1}{\sum_{j=1}^k N_j} \sum_{l=1}^{k} N_l \hat{m}_l \leq \frac{1}{\sum_{j=1}^k N_j} \sum_{l=1}^{k} N_l f^*_{N_l} \text{ a.s.} \tag{35}$$

Since $E[f^*_N] \leq f^*$ for all $N \in \mathbb{N}$, see, e.g., [20], it follows that $E[\hat{f}^*_{k+1}] \leq f^*$. Consequently, $\hat{f}^*_{k+1}$

is a lower bound on f^*, on average. In the case of a nonconvex problem, Assumption 1 typically only holds on subsets of X near locally optimal solutions with f_N^* replaced by the locally optimal values. Hence, in practice $\hat{f}_{k+1}^*$ may only be a probabilistic lower bound on a locally optimal value in the case of a nonconvex problem.

We now consider the asymptotic distribution of the lower bound on f_N^*.

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

$$N^{1/2}\left(\frac{f_N(A_N^n(x)) - \theta^n f_N(x)}{1-\theta^n} - f^*\right) \Rightarrow \inf_{x' \in X^*} Y(x') \tag{36}$$

as $N, n \to \infty$.

Proof: Since $A_N^n(x)$ is suboptimal, it follows that

$$f_N^* \geq \frac{f_N(A_N^n(x)) - \theta^n f_N(x)}{1-\theta^n} \geq \frac{f_N^* - \theta^n f_N(x)}{1-\theta^n} \text{ a.s.} \tag{37}$$

Hence,

$$\begin{aligned} & N^{1/2}(f_N^* - f^*) \\ \geq\ & \frac{N^{1/2}(f_N(A_N^n(x)) - f^*) - \theta^n N^{1/2}(f_N(x) - f^*)}{1-\theta^n} \\ \geq\ & \frac{N^{1/2}(f_N^* - f^*) - \theta^n N^{1/2}(f_N(x) - f^*)}{1-\theta^n} \text{ a.s.} \end{aligned} \tag{38}$$

The last expression tends to $\inf_{x' \in X^*} Y(x')$ as $n, N \to \infty$ with $\theta^n N^{1/2} \to 0$ by Proposition 1, the central limit theorem, and Slutsky's theorem (see, e.g., Exercise 25.7 of [5]). This result and Proposition 1 prove the theorem. □

When the sequence $\{f_{N_k}(x_i^k)\}_{i=0}^{n_k}$ is linearly progressing with rate of progression $\hat{\theta}_{k+1}$, then the minimization in (32) can be ignored. In this case, we see from Theorem 3 that $\hat{m}_k$ is approximately normally distributed with mean f^* and variance $\sigma^2(x^*)/N_k$ when N_k is large, n_k is large relatively to N_k, and $X^* = \{x^*\}$. In principle, a central limit theorem for triangular arrays could provide the asymptotic distribution of $\hat{f}_{k+1}^*$, as $k \to \infty$. However, we find such a result less useful as the number of stages before Algorithm 2 finds a near-optimal solution is typically rather small (e.g., 5-15). Hence, we proceed with a finite k and simply observe that if $\hat{m}_l$ is approximately normal for all stages $l = 1, 2, ..., k$, then $\hat{f}_{k+1}^*$ is approximately normally distributed with mean f^* and variance $\sigma^2(x^*)/\sum_{l=1}^k N_l$. We note that since the asymptotic distribution in (36) is independent of x, the dependence between the stages induced by the warm starting $x_0^l = x_{n_{l-1}}^{l-1}$ is insignificant for large number of iterations.

Combining the upper and lower bounds on f^* we obtain that $P[f(x_{n_k}^k) - f^* \leq \epsilon]$ is approximately bounded below by

$$\Phi\left(\frac{\hat{f}_{k+1}^* + \epsilon - f_{N^*}(x_{n_k}^k)}{\sqrt{\hat{\sigma}_{k+1}^2/\sum_{l=1}^k N_l + \hat{\sigma}_{k+1}^2/N^*}}\right), \tag{39}$$

where $\sigma^2(x^*)$ is estimated by $\hat{\sigma}^2_{k+1}$. Hence, if (39) is large, then it is likely that $x^k_{n_k} \in X^*_\epsilon$. The last solution of each stage in Algorithm 2 can be validated in this manner.

5 Computational Studies

There are obviously many alternatives to the sample-size policy induced by **S-SSCP**$_k$ in Step 1 of Algorithm 2. In this section we examine some that we believe are practical alternatives to the S-SSCP policy including the asymptotically optimal policy of the recent paper [26]. Specifically, we compare the computing time required to obtain a near-optimal solution by Algorithm 2 using different sample-size selection policies in Step 1. As mentioned in Section 1, stochastic programs may also be solved by algorithms *not* based on SAA. However, in this paper we do not compare across algorithmic frameworks and focus on efficient sample-size selection within SAA when applied to smooth stochastic programs.

We implement Algorithm 2 with Subroutine A in Matlab Version 7.4 on a laptop computer with 2.16 GHz processor, 2 GB RAM, and Windows XP operating system. We use one iteration of the projected gradient method with Armijo step size rule (see, e.g., p. 67 of [28]) as the algorithm map $A_N(\cdot)$. The quadratic direction finding problem in the projected gradient method is solved using LSSOL [8] as implemented in TOMLAB 7.0 [11].

In all computational tests, we use parameters $\alpha = 0.5$ and $\beta = 0.8$ in Armijo step size rule (see p. 67 of [28]) as well as exponential smoothing parameter $\phi = 1/3$ and tolerance $\epsilon_\theta = 0.0001$ in Subroutine A. 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 estimates of w_1 and w_2, see (34), are 3 and 1, respectively.

5.1 Numerical Examples

We consider the following five of problem instances for our numerical tests. The first three instances are constructed examples of **P** with known optimal solutions that allow us to examine different degree of ill-conditioning. The fourth problem instance arises in military and civilian search and rescue operations. The fifth instance arises in engineering design with multiple performance functions. The last problem instance illustrates that Algorithm 2 may be used even if $F(\cdot, \omega)$ is nonsmooth.

5.1.1 Problem Instances QUAD1, QUAD2, and QUAD3

Problem instances QUAD1, QUAD2, and QUAD3 are defined in terms of

$$F(x, \omega) = \sum_{i=1}^{20} a_i (x_i - b_i \omega_i)^2 \tag{40}$$

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 refer to the problem instance with $a_i = i$, $i = 1, 2, ..., 20$, by QUAD1, the problem instance with $a_i = 1 + 199(i - 1)/19$, $i = 1, 2, ..., 20$, by QUAD2, and $a_i = 1 + 1999(i - 1)/19$, $i = 1, 2, ..., 20$, by QUAD3. All these 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, QUAD1, QUAD2, and QUAD3 are strictly convex with identical unique global minimizer $x^* = (x_1^*, ..., x_{20}^*)'$, where $x_i^* = b_i/2$. The optimal value is $\sum_{i=1}^{20} a_i b_i^2/12$. We note that QUAD1, QUAD2, and QUAD3 are increasingly ill-conditioned. Even though solvable without SAA, we use these simple problem instances to illustrate our approach. For QUAD1, QUAD2, and QUAD3 we set $x_0^0 = 0 \in \mathbb{R}^{20}$ and use relative optimality tolerance 0.001, i.e., $\epsilon = 0.001p^*$ in Algorithm 2.

5.1.2 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 [46]). We generalize this problem and consider a random search effectiveness in cell i per time unit and minimize the expected probability of not detecting the target. 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 expected probability of not detecting the target is $f(x) = \mathbb{E}[F(x, \omega)]$, where

$$F(x, \omega) = \sum_{i=1}^{d} p_i e^{-\omega_i x_i}. \tag{41}$$

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 SAA, 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.

5.1.3 Problem Instance TRUSS

The last problem instance deals with the design of a truss structure with topology given in Figure 1. The truss is subject to a random load L in its mid-span. L is lognormally distributed with

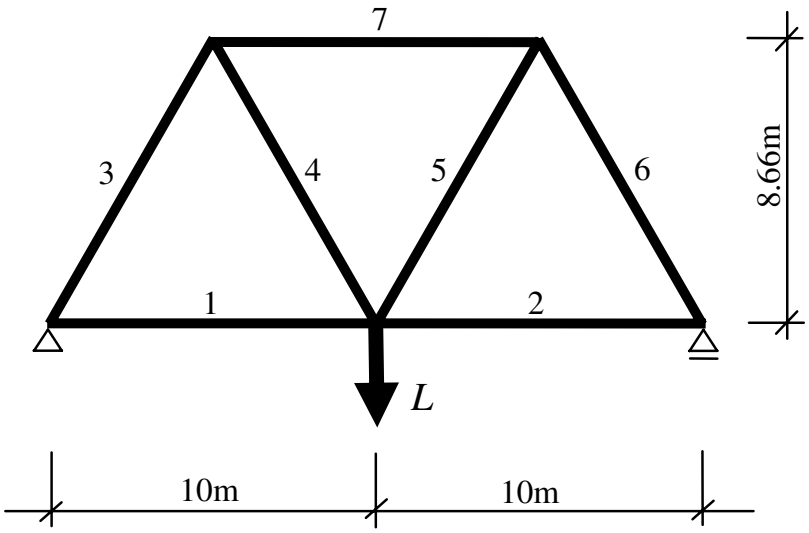

Figure 1: Design of Truss

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, 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 [36] for details). Using the approach in [36], see also [34], we find that this probability of failure can be approximated with high accuracy by

$$f(x) = \mathbb{E}[\max\{\rho, \max_{i=1,...,7}\{1 - \chi_8^2(r_i^2(x, \omega))\}\} \tag{42}$$

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 (42)

is of form (1) and is continuously differentiable under moderate assumptions [34]. However, the expression inside the brackets in (42) 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 10^7; see [30]. 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 $\omega \in \Omega$. A similar smoothing approach is used in [1, 47] for Conditional Value-at-risk minimization. This problem instance illustrates that also nonsmooth problem instances may be solved approximately by Algorithm 2.

The goal in this truss 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 note that this problem instance is not known to be convex. We use $x_0^0 = (3/7, ..., 3/7)' \in \mathbb{R}^7$ and $\epsilon = 0.05p^*$. In this problem instance, we increase the relative tolerance as the variability is rather high ($\sigma(x^*)/f^* \approx 10$).

5.2 Computational Results

We apply Algorithm 2 with different sample-size selection policies to the five problem instances. The measure of performance of the policies is the time required in Algorithm 2 until (39) exceeds the threshold 0.95. We do not carry out a rigorous analysis of such sequential tests, but find empirically that the coverage probabilities for Algorithm 2 with this stopping criterion is satisfactory. Specifically, Algorithm 2 stops after a stage k 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 QUAD1, which is well within the 5% indicated by the threshold 0.95. The stopping criterion yields satisfactory coverage also on the other problem instances and suffices for our purpose of comparing different policies. In view of this criterion, we select N^* so that the variability in $f_{N^*}(x_{n_k}^k)$ would tend not to prevent (39) from exceeding 0.95. That is, we set $N^* = \lceil (\hat{\sigma}_1 \Phi^{-1}(0.95)/(\epsilon/2))^2 \rceil$, which is the smallest sample size that ensures that (39) equals 0.95 when $f_{N^*}(x_{n_k}^k) - \hat{f}_{k+1}^* = \epsilon/2$ and there is no uncertainty in the lower bound, i.e., $\sum_{l=1}^{k} N_l$ "equals" infinity.

Table 1 gives average computing times in seconds over 10 independent runs of Algorithm 2, with standard deviations, when applied to QUAD1 (columns 4-5), QUAD2 (columns 6-7), and QUAD3 (columns 8-9). Each row represents a particular policy for determining (N_k, n_k). The third row gives the times when (N_k, n_k) is determined by the S-SSCP policy, rows 4-6 give times 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 7-9 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 10-12 give times for the same multiplicative policy as the previous rows except that $N_k = 2^{k-1} N_1$, $k = 2, 3, ...$ Rows 13-24 follow policies deduced from the recommendation in [26]. Specifically, from an N_1 given in column 3, $N_k = 1.1^{k-1} N_1$, $k = 2, 3, ...$, for rows 13-18 and $N_k = 1.5^{k-1} N_1$, $k = 2, 3, ...$, for rows 19-24. The number of iterations carried out at each stage is determined adaptively: the stage ends when $\|(\nabla^2 f_{N_k}(x_i^k))^{-1} \nabla f_{N_k}(x_i^k)\| \leq K/\sqrt{N_k}$. Column 2 of rows 13-24 gives the K used. The policies of

Policy			QUAD1		QUAD2		QUAD3	
Name	n_k	N_1	avg.	st.dev.	avg.	st.dev.	avg.	st.dev.
S-SSCP	adaptive	adaptive	182	98	240	110	223	98
Additive	5	$N^*/1000$	242	59	412	84	564	200
Additive	10	$N^*/1000$	434	213	394	111	611	343
Additive	20	$N^*/1000$	352	172	365	184	662	485
Mult. 1.5	5	$N^*/1000$	631	267	1351	1368	962	514
Mult. 1.5	10	$N^*/1000$	666	379	480	219	589	220
Mult. 1.5	20	$N^*/1000$	759	460	514	220	889	609
Mult. 2	5	$N^*/1000$	494	233	3958	3871	997	414
Mult. 2	10	$N^*/1000$	867	426	1112	879	1192	814
Mult. 2	20	$N^*/1000$	1080	1310	537	221	948	480
Mult. 1.1	$K = 1$	10	1434	170	2522	[4]	-	[0]
Mult. 1.1	$K = 1$	100	1039	229	2514	[9]	-	[0]
Mult. 1.1	$K = 1$	$N^*/1000$	501	200	1185	746	976	[1]
Mult. 1.1	$K = 0.1$	10	1521	318	2524	[3]	-	[0]
Mult. 1.1	$K = 0.1$	100	1015	323	1567	[3]	2963	[1]
Mult. 1.1	$K = 0.1$	$N^*/1000$	974	391	1301	[6]	2502	[3]
Mult. 1.5	$K = 1$	10	591	245	1690	[9]	3349	[1]
Mult. 1.5	$K = 1$	100	419	151	1187	[8]	1836	[2]
Mult. 1.5	$K = 1$	$N^*/1000$	305	79	1094	[8]	2521	[4]
Mult. 1.5	$K = 0.1$	10	864	427	2076	[6]	1143	[1]
Mult. 1.5	$K = 0.1$	100	841	778	1794	[6]	1186	[3]
Mult. 1.5	$K = 0.1$	$N^*/1000$	573	514	1925	[4]	2221	[2]

Table 1: Average and standard deviation of computing times (seconds) over ten runs of Algorithm 2 excluding the time of Step 1 when applied to QUAD1, QUAD2, and QUAD3. 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.

rows 13-24 are asymptotically optimal in a sense defined in [26]. We note that $N^*/1000$ is typically around 600 for QUAD1, QUAD2, and QUAD3, 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 anticipate that in real-world application of the proposed methodology the computing time required to approximately solve the surrogate sample-size control problem (Step 1 of Algorithm 2) will be negligible compared to the computing time of Step 3 in Algorithm 2. Hence, we exclude the time of Step 1 in the computing times reported in Table 1. For the present problem instances, the time of Step 1 is typically around 2.5 seconds and the number of stages is typically between 5 and 15.

We see from Table 1 that the S-SSCP policy is significantly better than the alternative ones for all there problem instances. The first additive policy (see row 4 of Table 1) appears to be reasonably efficient, but it requires roughly twice the computing time of the S-SSCP policy, on average. Other alternative policies may require as much as an order of magnitude more computing

Policy			SEARCH		TRUSS	
Name	n_k	N_1	avg.	st.dev.	avg.	st.dev.
S-SSCP	adaptive	adaptive	121	54	1593	512
Additive	5	$N^*/1000$	134	74	1992	2240
Additive	10	$N^*/1000$	253	158	2173	1369
Additive	20	$N^*/1000$	210	120	6875	6280
Mult. 1.5	5	$N^*/1000$	173	69	2661	2531
Mult. 1.5	10	$N^*/1000$	204	145	2751	1181
Mult. 1.5	20	$N^*/1000$	443	392	6756	7507
Mult. 2	5	$N^*/1000$	122	67	3303	2805
Mult. 2	10	$N^*/1000$	340	397	4557	4018
Mult. 2	20	$N^*/1000$	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 2: Average and standard deviation of computing times (seconds) over ten runs of Algorithm 2 excluding the time of Step 1 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.

time. The alternative policies deduced from the recommendation in [26] (see rows 13-24 in Table 1) perform comparable to the other alternative policies on QUAD1 (but still much worse than the S-SSCP policy). However, they result in extremely poor computing times for QUAD2 and QUAD3 with many runs not completed within one hour. It appears that the adaptive rule for determining the number of iterations for each stage in these policies tends to result in "over-solving" each stage. We also examined a fixed policy with $N_k = N^*/2$ and $n_k = 5$ for all k on QUAD1 (not reported in Table 1), but computing times exceeded 15000 seconds for all 10 runs.

In view of the results of Table 1, 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 [26], 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 recommendations perform worse than simple additive or multiplicative polices. On the other hand, the S-SSCP policy appears to be robust and it performs well even on ill-conditioned problems. In contrast to rigid additive and multiplicative polices, S-SSCP initially recommends many iterations per stage but reduces the number as the sample size is increased. When the sample size is large and the surrogate terminal state is almost satisfied, a cautious increase in sample size is recommended.

Table 2 gives similar result as Table 1 but for problem instances SEARCH and TRUSS. Each row represents a particular policy as described in Table 1 with the addition of three alternative policies using a fixed sample size $N_k = N^*/2$ and $n_k = 5$, 10, or 20 for all k, see rows 13-15 of Table 2. We do not examine the policies from [26] due to their poor performance in Table 1. On SEARCH,

the S-SSCP policy appears to be the fastest, but with averages over only 10 runs we cannot claim that the advantage over the best alternative policy is statistically significant. We notice, however, that the alternative policies often have significantly larger standard deviations than that associated with the S-SSCP policy. Consequently, the user of an alternative policy is exposed to a significant risk of having a long computing time even with a "good" choice of alternative policy.

On the problem instance TRUSS (see columns 6 and 7 of Table 2) the S-SSCP policy again outperforms the alternative policies with substantial margins. In this case, we also see that the alternative policies have significantly larger standard deviations (coefficient of variations of roughly 1) than that of the S-SSCP policy (coefficient of variation of roughly 1/3). Hence, the user of an alternative policy not only should expect a longer computing time on average as compared to the S-SSCP policy, but also the possibility of much longer times. We observe that the best alternative policy for SEARCH (see row 10 of Table 2) 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.

6 Conclusions

We consider the solution of smooth stochastic programs by sample average approximations and formulate 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 approximately solve the 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 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 direction to a technical result.

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] 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.

[3] G. Bayraksan and D.P. Morton. A sequential sampling procedure for stochastic programming. *Operations Research*, to appear, 2009.

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

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

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

[7] 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.

[8] 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.

[9] 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.

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

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

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

[13] 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.

[14] 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.

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

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

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

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

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

[20] 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.

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

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

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

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

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

[26] R. Pasupathy. On choosing parameters in retrospective-approximation algorithms for stochastic root finding and simulation optimization. *Operations Research*, to appear, 2009.

[27] 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.

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

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

[30] 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.

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

[32] J. O. Royset. Optimality functions in stochastic programming. *Available at http://faculty.nps.edu/joroyset/docs/Royset_optimfcn.pdf*, 2009.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

[37] 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.

[38] 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.

[39] 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.

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

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

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

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

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

[45] B. Verweij, S. Ahmed, A.J. Kleywegt, G. Nemhauser, and A. Shapiro. The sample average approximation method applied to stochastic routing problems: A computational study. *Computational Optimization and Applications*, 24:289–333, 2003.

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

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

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