Available online at www.sciencedirect.com
ScienceDirect
Journal of Computational and Applied Mathematics 217 (2008) 301–310
JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS
www.elsevier.com/locate/cam

Efficient sample sizes in stochastic nonlinear programming

E. Polak[a], J.O. Royset[b,*]

[a] *Department of Electrical Engineering & Computer Sciences, University of California, Berkeley, CA 94720, USA*
[b] *Operations Research Department, Naval Postgraduate School, Monterey, CA 93943, USA*

Received 1 April 2006; received in revised form 25 July 2006

Abstract

We consider a class of stochastic nonlinear programs for which an approximation to a locally optimal solution is specified in terms of a fractional reduction of the initial cost error. We show that such an approximate solution can be found by approximately solving a sequence of sample average approximations. The key issue in this approach is the determination of the required sequence of sample average approximations as well as the number of iterations to be carried out on each sample average approximation in this sequence. We show that one can express this requirement as an idealized optimization problem whose cost function is the computing work required to obtain the required error reduction. The specification of this idealized optimization problem requires the exact knowledge of a few problems and algorithm parameters. Since the exact values of these parameters are not known, we use estimates, which can be updated as the computation progresses. We illustrate our approach using two numerical examples from structural engineering design.
Published by Elsevier B.V.

MSC: 90C15; 90C30; 90C90

Keywords: Stochastic optimization; Sample average approximations; Diagonalization; Reliability-based optimal design

1. Introduction

In many practical decision making situations, uncertainty about future events as well as present status is significant and must be accounted for in any optimization model of the system considered. One approach is to optimize the system on the average, which may lead to a stochastic nonlinear programming problem in the following general form:

$$\mathbf{P}: \min_{x \in X} g(x), \tag{1}$$

where $x \in X \subset \mathbb{R}^n$ is a vector of continuous decision variables and

$$g(x) = EG(x, \omega). \tag{2}$$

Here, E is the expectation corresponding to a probability space $(\Omega, \mathscr{F}, P)$, where Ω is a sample space, $\mathscr{F}$ is a σ-algebra, and P is a probability measure. Furthermore, $G : \mathbb{R}^n \times \Omega \to \mathbb{R}$ is a real-valued function, continuously differentiable

* Corresponding author. Tel.: +1 831 656 2578; fax: +1 831 656 2595.
E-mail addresses: polak@eecs.berkeley.edu (E. Polak), joroyset@nps.edu (J.O. Royset).

0377-0427/$ - see front matter Published by Elsevier B.V.
doi:10.1016/j.cam.2007.02.014

with respect to its first argument for all $\omega \in \Omega$. We assume that P is completely known. Furthermore, we assume that for every x in a sufficiently large set, $G(x, \cdot)$ is $\mathscr{F}$-measurable and P-integrable, and hence $g(x)$ is well defined. For a broad, in-depth presentation of stochastic programming we refer to [14].

For simplicity, we assume that the feasible region is defined by

$$X = \{x \in \mathbb{R}^n \mid f_j(x) \leqslant 0,\ j \in \mathbf{J}\}, \tag{3}$$

where $f_j : \mathbb{R}^n \to \mathbb{R}$, $j \in \mathbf{J} = \{1, 2, \ldots, J\}$, are deterministic, continuously differentiable constraint functions. However, extensions to other feasible regions including those defined in terms of expectations are possible, e.g., by using penalty functions.

Typically, the expectation $EG(x, \omega)$ cannot be computed exactly because P is a continuous distribution or computations are impractical because P is a discrete distribution with a finite, but large number of possible realizations. For this reason, Monte-Carlo sampling-based approaches have emerged. In the context of **P**, a Monte-Carlo approach replaces $g(x)$ by its sample average leading to the following sample average approximation:

$$\mathbf{P}_N : \min_{x \in X} g_N(x), \tag{4}$$

where

$$g_N(x) = \sum_{j=1}^{N} G(x, \omega_j)/N \tag{5}$$

and $\omega_1, \omega_2, \ldots, \omega_N$ is a random sample. Given a sample, $\mathbf{P}_N$ is a deterministic optimization problem, which can be solved by various nonlinear programming algorithms.

Several asymptotic results are available for sample average approximations under weak assumptions including the fact that minimizers and minimum values of the sequence of problems $\{\mathbf{P}_N\}_{N=1}^{\infty}$ converge with probability one to minimizers and the minimum value of **P**, respectively, see, e.g., [5,6,9,14,15]. Rate of convergence results can also be found under additional assumptions, see [14, Section 6.2.2]. In view of these results, one approach for solving **P** is to select a "sufficiently large" sample size, say N^*, generate a sample of size N^*, and solve $\mathbf{P}_{N^*}$ using an appropriate optimization algorithm. The quality of the obtained solution can then be assessed using one of the techniques found in, e.g., [14, Section 6.4], which may involve solving $\mathbf{P}_{N^*}$ multiple times for different independent samples.

Even though rate of convergence results provide some guidance, it can be difficult to select an appropriate sample size which balances accuracy with computational tractability. This is further complicated if a "diagonalization" scheme is used. In a diagonalization scheme, **P** is not (approximately) solved by solving one sample average approximation, but by solving it in stages, each stage consisting of performing a finite number of solver iterations on a problem $\mathbf{P}_{N_i}$, $i = 1, 2, \ldots, s$, where $N_1, N_2, \ldots, N_s$ is an increasing sequence of sample sizes, and s is the number of stages. In this approach, the result of the last iteration at stage i is used as a warm-start for the calculations at the next stage $i + 1$. Intuition and computational evidence suggest that a diagonalization scheme can reduce the overall computing time since it uses coarse approximations, characterized by small sample sizes, when far from a local solution, and improves the quality of the approximation (i.e., increases the sample size), only as a local solution of **P** is approached.

However, the problem of selecting appropriate sample sizes and deciding when to switch to the next stage is nontrivial. Particularly, since these decisions are bound to have a serious impact on the total computing time. In [13], using ideas found in semi-infinite optimization, we preselect the sizes $\{N_i\}$ of the stages, i.e., problems $\{\mathbf{P}_{N_i}\}$ to be used, and present a closed-loop, feedback test for determining when to construct the next stage and to switch to it. According to this test, when the algorithm being applied to $\mathbf{P}_{N_i}$ starts to stall, making only very small cost-reductions, it is time to switch to $\mathbf{P}_{N_{i+1}}$. In this test, the critical cost-reduction depends on the index i. Assuming that the number of stages is allowed to be infinite, the test in [13] ensures that a sufficient number of solver iterations are carried out at each stage to guarantee asymptotic convergence of the overall algorithm to a local solution of **P**.

In this paper we do not consider the issue of asymptotic convergence, but rather the problem of obtaining an approximate solution that is specified in terms of a fractional reduction in cost error from an initial cost error. The

feedback test proposed in [13] is not useful for this task. Instead, we follow the process proposed in [4], which consists of formulating an auxiliary optimization problem, whose cost function is the total work needed to obtain a solution of required accuracy, and whose constraint is that the required cost reduction be achieved. The variables in this auxiliary problem are (i) the number of stages, s, to be used, (ii) the sample size N_i to be used in stage i, $i = 1, 2, \ldots, s$, and (iii) the number of solver iterations n_i to be used in stage i. While the number of stages s has to be treated as an integer variable, the variables N_i and n_i can be treated as continuous variables and rounded at the end of their optimization. In practice, it turns out that the optimal number of stages s^* hardly ever exceeds 10, with 3–7 being a most likely range for s^*. Incidentally, if one assigns the number of stages to be $s > s^*$, and then solve the reduced auxiliary optimization problem for the N_i and n_i, the optimal solution will consist of several N_i being equal, so that the total number of *distinct* stages is s^*.

The auxiliary problem depends on a sampling-error bound, on the initial cost error, and on the rate of convergence of the solver. All of these may have to be estimated. As a result, it may be presumptuous to call the solution of the auxiliary optimization problem an "optimal strategy," and hence we will call it an "efficient strategy." As we will see from our numerical results, despite the use of estimated quantities, the efficient strategy is considerably more effective than the obvious alternatives.

In Section 2, we derive the auxiliary optimization problem for determining the efficient diagonalization strategy. Section 3 describes how parameters in the auxiliary problem can be estimated. The overall scheme is described in Section 4. We present two numerical examples in Section 5.

2. Efficient diagonalization

We begin by deriving the auxiliary optimization problem. The auxiliary optimization problem is similar to the one in [4], but the following derivation of the model as well as its implementation is somewhat simplified.

First we use exact penalization to convert $\mathbf{P}$ into an unconstrained min–max problem. For a given $\pi > 0$, we define

$$\psi(x) = g(x) + \pi \| f(x)_+ \|_\infty, \tag{6}$$

and

$$\psi_N(x) = g_N(x) + \pi \| f(x)_+ \|_\infty, \tag{7}$$

where $\| f(x)_+ \|_\infty = \max\{0, f_1(x), f_2(x), \ldots, f_J(x)\}$. For a given penalty $\pi > 0$, let

$$\tilde{\mathbf{P}} : \min_{x \in \mathbb{R}^n} \psi(x). \tag{8}$$

If $\mathbf{P}$ is calm (see, e.g., [2,3]) and π is sufficiently large, then $x \in \mathbb{R}^n$ is a local minimizer of $\tilde{\mathbf{P}}$ if and only if it is a local minimizer of $\mathbf{P}$. Similarly,

$$\tilde{\mathbf{P}}_N : \min_{x \in \mathbb{R}^n} \psi_N(x) \tag{9}$$

is equivalent to $\mathbf{P}_N$ for sufficiently large π. An appropriate π can be selected using well-known techniques such as the one in [10, Section 2.7.3]. The implementation of exact penalty algorithms is beyond the scope of this paper and we assume in the following that a sufficiently large $\pi > 0$ has been determined.

Throughout this paper, we assume that each sample point is independently generated from P and that sample points are reused at later stages, i.e., for all $i = 2, 3, \ldots, s$, the sample at stage i consists of the N_{i-1} sample points at stage $i-1$ and $N_i - N_{i-1}$ new, independent sample points. Suppose that a random sample $\omega_1, \omega_2, \ldots, \omega_{N_s}$ has been realized and that $g_{N_i}(x)$ is computed using this sample for all $x \in \mathbb{R}^n$ and $i = 1, 2, \ldots, s$. Hence, we can argue deterministically in the following paragraphs.

We denote the positive integers by $\mathbb{N} = \{1, 2, 3, \ldots\}$. To construct an optimization model for determining the number of stages as well as the sample size at each stage and number of solver (min–max algorithm) iterations to be performed at each stage, we introduce the following assumptions. Suppose that the min–max algorithm applied to $\{\tilde{\mathbf{P}}_{N_i}\}_{i=1}^s$ is defined

by a linearly convergent algorithm map $A : \mathbb{R}^n \times C(\mathbb{R}^n, \mathbb{R}) \to \mathbb{R}^n$ to compute the sequences $\{x_j^i\}_{j=0}^{n_i}$, $i = 1, 2, \ldots, s$, where $n_i \in \mathbb{N}$ is the number of iterations at stage i, i.e., $x_{j+1}^i = A(x_j^i, \psi_{N_i})$ for all $j = 0, 1, \ldots, n_i - 1$ and $i = 1, 2, \ldots, s$. To make use of "warm" starts, we set $x_0^i = x_{n_{i-1}}^{i-1}$. We assume that the rate of convergence of the algorithm map is the same for the entire family $\{\tilde{\mathbf{P}}_{N_i}\}_{i=1}^s$, i.e., there exists a $\theta \in (0, 1)$ such that for all $x \in \mathbb{R}^n$ and $i = 1, 2, \ldots, s$

$$\psi_{N_i}(A(x, \psi_{N_i})) - \psi_{N_i}^* \leqslant \theta(\psi_{N_i}(x) - \psi_{N_i}^*), \tag{10}$$

where $\psi_{N_i}^*$ is the optimal value for the problem $\tilde{\mathbf{P}}_{N_i}$. Next, assume that for all $N \in \mathbb{N}$, $N \leqslant N_s$ and for all x in a sufficiently large subset set of $\mathbb{R}^n$ containing $\{x_j^i\}_{j=0}^{n_i}$, $i = 1, 2, \ldots, s$ and the optimal solutions of $\tilde{\mathbf{P}}$ and $\tilde{\mathbf{P}}_N$, $N \leqslant N_s$, the sampling error is given by

$$|\psi_N(x) - \psi(x)| \leqslant \Delta(N), \tag{11}$$

where, $\Delta : \mathbb{N} \to (0, \infty)$ is a strictly decreasing function with $\Delta(N) \to 0$, as $N \to \infty$. We return to the form of $\Delta(\cdot)$ below, but for now we only assume that such a function exists.

Let ψ^* and $x^* \in \mathbb{R}^n$ be the optimal value and an optimal solution of $\tilde{\mathbf{P}}$, respectively. Also, let $x_N^* \in \mathbb{R}^n$ be an optimal solution of $\tilde{\mathbf{P}}_N$, i.e., $\psi_N^* = \psi_N(x_N^*)$. Then, in view of (11) we have that

$$\psi^* \leqslant \psi(x_N^*) \leqslant \psi_N(x_N^*) + \Delta(N) = \psi_N^* + \Delta(N), \tag{12}$$

$$\psi_N^* \leqslant \psi_N(x^*) \leqslant \psi(x^*) + \Delta(N) = \psi^* + \Delta(N) \tag{13}$$

for all $N \leqslant N_s$.

For any stage $i = 1, 2, \ldots, s$, we define the cost-to-go after the last iteration of the ith stage by

$$e_i = \psi(x_{n_i}^i) - \psi^*. \tag{14}$$

Also let $e_0 = \psi(x_0^1) - \psi^*$. Using (11)–(13) and (10), we obtain that for all $i = 1, 2, \ldots, s$,

$$e_i \leqslant \psi_{N_i}(x_{n_i}^i) - \psi_{N_i}^* + 2\Delta(N_i) \tag{15}$$

$$\leqslant \theta^{n_i}[\psi_{N_i}(x_0^i) - \psi_{N_i}^*] + 2\Delta(N_i) \tag{16}$$

$$\leqslant \theta^{n_i}[\psi(x_{n_{i-1}}^{i-1}) - \psi^*] + 4\Delta(N_i) \tag{17}$$

$$\leqslant \theta^{n_i} e_{i-1} + 4\Delta(N_i). \tag{18}$$

Hence,

$$e_s \leqslant e_0 \theta^{k_0(s)} + 4 \sum_{i=1}^{s} \theta^{k_i(s)} \Delta(N_i), \tag{19}$$

where $k_i(s) = \sum_{l=i+1}^{s} n_l$ if $i < s$ and $k_i(s) = 0$ if $i = s$. As the number of stages s increases to infinity, the first term in (19) vanishes while the second expression consists of an increasing number of terms. However, as the following theorem shows, the cost-to-go is guaranteed to vanish as the number of stages tends to infinity.

Theorem 2.1. *Consider a sequence $\{N_i\}_{i=1}^{\infty}$ of sample sizes, with $N_i \to \infty$, as $i \to \infty$, and sequences $\{x_j^i\}_{j=0}^{n_i}$, with $n_i \in \mathbb{N}$, $i = 1, 2, \ldots, s$, generated by the algorithm map $A : \mathbb{R}^n \times C(\mathbb{R}^n, \mathbb{R}) \to \mathbb{R}^n$ using the recursion $x_{j+1}^i = A(x_j^i, \psi_{N_i})$ for all $j = 0, 1, \ldots, n_i - 1$ and $i = 1, 2, \ldots, s$, with $x_0^i = x_{n_{i-1}}^{i-1}$, $i = 2, 3, \ldots, s$, and $x_0^1 \in \mathbb{R}^n$ given.*

Suppose that the following hold almost surely:

(i) *There exists a* $\theta \in (0, 1)$ *such that for all* $x \in \mathbb{R}^n$ *and* $i \in \mathbb{N}$, (10) *holds*.
(ii) *There exists a strictly decreasing function* $\Delta : \mathbb{N} \to (0, \infty)$ *such that* $\Delta(N) \to 0$, *as* $N \to \infty$, *and*

$$|\psi_{N_i}(x) - \psi(x)| \leqslant \Delta(N_i), \tag{20}$$

for all $x \in \mathbb{R}^n$ *and* $i \in \mathbb{N}$.

Then, $e_s \to 0$, *as* $s \to \infty$, *almost surely*.

Proof. We first observe that any sequence $\{a_i\}_{i=0}^{\infty}$ constructed by the recursion $a_i = \theta a_{i-1} + b$, with $\theta \in (0, 1)$ and $b \geqslant 0$, converges to $b/(1-\theta)$, as $i \to \infty$.

Since $\Delta(N_i) \to 0$, as $i \to \infty$, there exists a strictly decreasing sequence $\{d_i\}_{i=1}^{\infty}$ such that $d_i \to 0$, as $i \to \infty$, and $d_i \geqslant 4\Delta(N_i)$ for all $i \in \mathbb{N}$.

We now construct the collection of sequences $\{e_s^j\}_{s=s_j}^{\infty}$, $j \in \mathbb{N}$, in the following manner: Let $s_1 = 0$, $e_{s_1}^1 = e_{s_1} = e_0$, and $e_s^1 = \theta e_{s-1}^1 + d_{s_1+1}$ for all $s \geqslant s_1 + 1$. Since $e_s^1 \to d_{s_1+1}/(1-\theta)$, as $s \to \infty$, there exists an integer $s_2 > s_1$ such that $e_s^1 \leqslant 2d_{s_1+1}/(1-\theta)$ for all $s \geqslant s_2$. Hence, we have constructed s_1, $\{e_s^1\}_{s=s_1}^{\infty}$, and s_2.

In a similar manner, we construct $\{e_s^2\}_{s=s_2}^{\infty}$ and s_3: Let $e_{s_2}^2 = e_{s_2}^1$ and $e_s^2 = \theta e_{s-1}^2 + d_{s_2+1}$ for all $s \geqslant s_2 + 1$. There exists an integer $s_3 > s_2$ such that $e_s^2 \leqslant 2d_{s_2+1}/(1-\theta)$ for all $s \geqslant s_3$. Hence, we have constructed $\{e_s^2\}_{s=s_2}^{\infty}$, and s_3.

We obtain $\{e_s^j\}_{s=s_j}^{\infty}$, $j = 3, 4, \ldots$, by repeating this processes. Consequently,

$$e_s^j \leqslant \frac{2d_{s_j+1}}{1-\theta} \tag{21}$$

for all $s \geqslant s_{j+1}$ and $j \in \mathbb{N}$.

Since the number of iterations at each stage $n_i \geqslant 1$ for all $i \in \mathbb{N}$, $\theta \geqslant \theta^{n_i}$ for all $i \in \mathbb{N}$. Hence, it follows from (18) and the construction of $\{e_s^j\}_{s=s_j}^{\infty}$, $j \in \mathbb{N}$, that $e_s \leqslant e_s^j$ for all $s \geqslant s_j$ and $j \in \mathbb{N}$. This result, together with (21), gives that

$$e_s \leqslant \frac{2d_{s_j+1}}{1-\theta} \tag{22}$$

for all $s \geqslant s_{j+1}$ and $j \in \mathbb{N}$. Since $d_{s_j+1} \to 0$, as $j \to \infty$, the conclusion follows. □

Assumption (ii) in Theorem 2.1 is satisfied under moderate assumptions using a uniform Law of the Iterated Logarithm (see, e.g., [7, p. 217], and [13]). In this case, $\Delta(N) = \sqrt{(\log \log N)/N}$ for sufficiently large N. We also note that $\{N_i\}_{i=1}^{\infty}$ is not required to be strictly decreasing for the conclusion in Theorem 2.1 to hold.

Theorem 2.1 shows that diagonalization schemes can be constructed with asymptotic convergence. This is a valuable result, but we aim to determine efficient diagonalization schemes, i.e., schemes that minimize the computing time to reach a specific reduction in cost from an initial value.

To be able to construct efficient diagonalization schemes we need to quantify the computational effort associated with one iteration of the algorithm map $A(\cdot, \psi_N)$ as a function of the sample size N. Suppose that this computational effort is given by $w(N)$ for any $x \in \mathbb{R}^n$, where $w : \mathbb{N} \to (0, \infty)$ is some function. We are now ready to present the efficient diagonalization problem.

Given an initial cost-to-go $e_0 > 0$ and an $\varepsilon \in (0, 1)$, we seek to determine the number of stages $s \in \mathbb{N}$ as well as sample sizes $N_i \in \mathbb{N}$ and numbers of iterations n_i at each stage i, $i = 1, 2, \ldots, s$, such that the computational effort to reach a cost-to-go of εe_0 is minimized. In view of (19), this optimization problem takes the following form:

$$\mathbf{D}(e_0, \varepsilon) : \min_{s \in \mathbb{N}} \min_{n_i, N_i} \left\{ \sum_{i=1}^{s} n_i w(N_i) \,\middle|\, e_0 \theta^{k_0(s)} + 4 \sum_{i=1}^{s} \theta^{k_i(s)} \Delta(N_i) \leqslant \varepsilon e_0, \right.$$
$$\left. N_{i+1} \geqslant N_i, i = 1, 2, \ldots, s-1, n_i, N_i \in \mathbb{N}, i = 1, 2, \ldots, s \right\}. \tag{23}$$

The estimation of the parameters defining problem $\mathbf{D}(e_0, \varepsilon)$ is discussed in the next section. For lack of a better phrase, we will refer to this process as the implementation of $\mathbf{D}(e_0, \varepsilon)$.

3. Implementation of efficient diagonalization problem

The efficient diagonalization problem $\mathbf{D}(e_0, \varepsilon)$ involves the work and sampling-error functions $w(\cdot)$ and $\Delta(\cdot)$ as well as the rate of convergence parameter θ and the initial cost-to-go $e_0 = \psi(x_0^1) - \psi^*$. All these quantities must be determined before $\mathbf{D}(e_0, \varepsilon)$ can be solved. We deal with these issues one at a time.

In view of (5), the computing effort required to evaluate $g_N(\cdot)$ and its gradient grows linearly in N. Hence, the work associated with one iteration of the algorithm map $A(\cdot, \psi_N)$ is proportional to N and the work function $w(N) = N$.

The sampling error $\Delta(N)$ can be determined using the Law of the Iterated Logarithm as indicated earlier. However, $\sqrt{(\log \log N)/N}$ is a pessimistic estimate of the sampling error "typically" experienced. Since our goal is to determine efficient number of stages, sample sizes, and numbers of iterations, it appears to be more reasonable to assume that the sampling error is proportional to $1/\sqrt{N}$ as proposed by classical estimation theory: For a given $x \in \mathbb{R}^n$, it follows under weak assumption from the Central Limit Theorem that $\psi_N(x)$ is approximately normally distributed with mean $\psi(x)$ and variance $\sigma(x)^2/N$ for large N, where $\sigma(x)^2 = Var[G(x, \omega)]$. Hence, for sufficiently large N,

$$P[|\psi_N(x) - \psi(x)| \leqslant \Delta(N)] \geqslant 0.95, \tag{24}$$

when $\Delta(N) = 1.96\sigma(x)/\sqrt{N}$. This error expression appears to be more appropriate, and we set $\Delta(N) = 1.96 \sup_{x\in\mathbb{R}^n} \sigma(x)/\sqrt{N}$ in the following. Usually, $\sup_{x\in\mathbb{R}^n} \sigma(x)$ is unknown and must be estimated. The same is true for θ and e_0.

We determine $\sigma(x)$, θ, and e_0 in an estimation phase consisting of n_0 iterations of the algorithm map $A(\cdot, \psi_{N_0})$ applied to $\mathbf{P}_{N_0}$, with N_0 being a small sample size. Let $\{x_j^0\}_{j=0}^{n_0}$ be the iterates computed in the estimation phase. Each time $g_{N_0}(x)$ is computed, the corresponding variance $\sigma(x)^2$ is estimated by its unbiased estimator

$$\hat{\sigma}(x)^2 = \sum_{j=1}^{N_0} (G(x, \omega_j) - g_N(x))^2/(N_0 - 1). \tag{25}$$

Hence, we approximate $\sup_{x\in\mathbb{R}^n} \sigma(x)$ by $\hat{\sigma} = \max_{j=0,1,\ldots,n_0} \hat{\sigma}(x_j^0)$.

The rate of convergence parameter θ is estimated by the solution of the following least-squares problem, where the optimal value $\psi_{N_0}^*$ of $\tilde{\mathbf{P}}_{N_0}$ is also estimated:

$$\min_{\hat{\theta},\hat{\psi}} \sum_{j=0}^{n_0} [(\hat{\psi} + (\psi_{N_0}(x_0^0) - \hat{\psi})\hat{\theta}^j) - \psi_{N_0}(x_j^0)]^2. \tag{26}$$

Then, we estimate θ by $\hat{\theta}$ and $\psi_{N_0}^*$ by $\hat{\psi}$. Finally, we estimate the initial cost-to-go $e_0 = \psi(x_0^1) - \psi^*$ by $\hat{e}_0 = \psi_{N_0}(x_0^0) - \hat{\psi}$.

We have now established procedures for estimating all the unknown quantities in $\mathbf{D}(e_0, \varepsilon)$. $\mathbf{D}(e_0, \varepsilon)$ is a nonlinear integer program that appears difficult to solve directly, but this fact can be circumvented by the following observations.

First, the restriction of $\mathbf{D}(e_0, \varepsilon)$ obtained by fixing $s \in \mathbb{N}$ to a number in the range 5–10 tends to be insignificant since more than 5–10 stages is rarely advantageous and fewer than 5–10 stages is still effectively allowed in the model by setting $N_i = N_{i+1}$ for some i. Second, N_i, and to some extent also n_i, tend to be large integers. Hence, a continuous relaxation with rounding of the optimal solutions to the nearest integers is justified. In view of these observations, $\mathbf{D}(e_0, \varepsilon)$ can be solved approximately using a standard nonlinear programming algorithm.

4. Overall algorithm

We now summarize our approach and discuss how the efficient diagonalization problem can be integrated in an algorithm for solving $\mathbf{P}$. As indicated above, the process of solving the efficient diagonalization problem must be preceded by an estimate phase where parameters are determined. This leads to the following overall algorithm for solving $\mathbf{P}$ approximately.

Algorithm for solving P approximately.

Parameters. Number of iterations in estimation phase $n_0 \in \mathbb{N}$, sample size in estimation phase $N_0 \in \mathbb{N}$, maximum number of stages s, and constraint penalty $\pi > 0$.

Data. Required fractional reduction in cost $\varepsilon > 0$, initial point $x_0^0 \in \mathbb{R}^n$, and a set $\{\omega_1, \omega_2, \ldots\}$ of independent sample points from P.

Step 0. Compute variance estimate $\hat{\sigma}(x_0^0)^2$ using (25).

Step 1. For $j = 0$ to $n_0 - 1$, compute $x_{j+1}^0 = A(x_j^0, \psi_{N_0})$ and the variance estimate $\hat{\sigma}(x_{j+1}^0)^2$ using (25).

Step 2. Compute the global estimate $\hat{\sigma} = \max_{j=0,1,\ldots,n_0} \hat{\sigma}(x_j^0)$.

Step 3. Determine $\hat{\theta}$ and $\hat{\psi}$ as the optimal solution of (26).

Step 4. Set $\hat{\Delta}(N) = 1.96\hat{\sigma}/\sqrt{N}$, and determine n_i and N_i by solving

$$\min_{n_i, N_i} \left\{ \sum_{i=1}^{s} n_i N_i \,\middle|\, \hat{e}_0 \hat{\theta}^{k_0(s)} + 4 \sum_{i=1}^{s} \hat{\theta}^{k_i(s)} \hat{\Delta}(N_i) \leqslant \varepsilon \hat{e}_0,\ N_{i+1} \geqslant N_i,\ i = 1, 2, \ldots, s-1,\ n_i,\ N_i \geqslant 1,\ i = 1, 2, \ldots, s \right\}. \tag{27}$$

Step 5. For $i = 1$ to s, perform

Sub-step 5.1. Set $x_0^i = x_{n_{i-1}}^{i-1}$.

Sub-step 5.2. For $j = 0$ to $n_i - 1$, compute $x_{j+1}^i = A(x_j^i, \psi_{N_i})$.

The proposed algorithm consists of three phases: estimation of parameters (Steps 0–3), solution of efficient diagonalization problem (Step 4), and main iterations (Step 5). This represents the simplest implementation of our idea. Alternatively, we can adopt a moving-horizon approach, where Step 5 is completed only for $i = 1$, followed by Step 4, then by Step 5 for $i = 1$ again, followed by Step 4, etc. Hence, the diagonalization plan is re-optimized after each stage, which may lead to an improved plan. With re-optimization, it is also possible to re-compute $\hat{\sigma}$, using all previous iterates, as well as $\hat{\theta}$ and $\hat{\psi}$. Other implementations can also be imagined. In the following numerical study, we adopt the simple implementation described above.

5. Numerical study

We illustrate our approach using two numerical examples, both of which arise in structural engineering design. The examples are implemented in Matlab 7.0 [8] on a 2.8 GHz PC running Microsoft Windows 2000. We use one iteration of the Pshenichnyi–Pironneau–Polak Min–Max Algorithm (see [10, Section 2.4.1]) as the algorithm map $A(\cdot, \cdot)$.

The first example arises in optimal design of a short structural column with a rectangular cross section of dimensions $x_1 \times x_2$. Hence, $x = (x_1, x_2) \in \mathbb{R}^2$ is the design vector. The column is subjected to bi-axial bending moments V_1 and V_2, which, together with the yield strength V_3 of the material, are considered to be independent, lognormally distributed random variables. The column is also subject to a deterministic axial force a_f. This gives rise to a failure probability

$$p(x) = P_V[\{h(x, V) \leqslant 0\}], \tag{28}$$

where P_V is the probability distribution of the random vector $V = (V_1, V_2, V_3)$ and $h : \mathbb{R}^2 \times \mathbb{R}^3 \to \mathbb{R}$ is a limit-state function defined by

$$h(x, V) = 1 - \frac{4V_1}{x_1 x_2^2 V_3} - \frac{4V_2}{x_1^2 x_2 V_3} - \left(\frac{a_f}{x_1 x_2 V_3} \right)^2. \tag{29}$$

Now, let $\Omega = \{\omega \in \mathbb{R}^3 \,|\, \|\omega\| = 1\}$ and P be the uniform distribution on the three-dimensional hypersphere. In [13], we show that under fairly general assumptions the failure probability can be approximated by the expression

$$p_a(x) = E[G(x, \omega)], \tag{30}$$

to arbitrarily high accuracy for large values of the parameter $a > 0$ ($a = 6.5$ suffices in this example), where

$$G(x, \omega) = \max\{1 - \chi^2(r^2(x, \omega)), 1 - \chi^2(a^2)\}, \tag{31}$$

$\chi^2(\cdot)$ is the Chi-square cumulative distribution function with 3 degrees of freedom,

$$r(x, \omega) = \begin{cases} \min\limits_{r}\{r|\tilde{h}(x, r\omega) \leqslant 0, r \geqslant 0\} & \text{if} \quad \{r|\tilde{h}(x, r\omega) \leqslant 0, r \geqslant 0\} \neq \emptyset, \\ \infty & \text{otherwise}, \end{cases} \tag{32}$$

and $\tilde{h}(\cdot, \cdot)$ is a standardized version of $h(\cdot, \cdot)$. We note that estimation of (30) tends to require fewer samples than estimation of (28) in the case of small probabilities. This is the case since (28) is estimated by an average of zeros and ones. However, in general the advantage for (30) diminishes as the dimension of the sample space Ω increases.

We seek a design of the column which satisfies the constraints defined by $f_1(x) = -x_1$, $f_2(x) = -x_2$, $f_3(x) = x_1/x_2 - 2$, $f_4(x) = 0.5 - x_1/x_2$, $f_5(x) = x_1x_2 - 0.175$, and minimize $p_a(x)$. Hence, the problem is in the form of $\mathbf{P}$, but with the integrand $G(\cdot, \omega)$ nonsmooth. Under weak additional assumptions, it follows from [13] that $p_a(x)$ is continuously differentiable. Hence, for moderately large N, the sample average $\sum_{j=1}^{N} G(x, \omega_j)/N$ of $p_a(x)$ is, for practical purposes, effectively smooth. For this reason, we ignore the nonsmoothness of the sample average function in this, as well as the next example. No detrimental behavior of the Pshenichnyi–Pironneau–Polak Min–Max Algorithm was observed because of this simplification. Alternatively, we could have applied the entropy technique from [11] to smooth the sample average, but we opted for the simpler approach, which equally well illustrates our efficient diagonalization strategy.

The algorithm parameters were selected to be $n_0 = 25$, $N_0 = 50$, $s = 5$, and $\pi = 2$. We note that $\pi = 2$ suffices to ensure feasibility in $\mathbf{P}$. Finally, the required fractional reduction in cost was $\varepsilon = 0.01$ and the initial point was chosen to be $x_0^0 = (\sqrt{0.175}, \sqrt{0.175})$.

Our proposed algorithm yielded a diagonalization strategy of three stages with 25, 8, and 8 iterations, with sample sizes 50, 251, and 1621, respectively, which was executed in 458 s. Note that the computing time includes the estimation phase (30 s) and the solution time of the efficient diagonalization problem (3 s).

For comparison, we also adapted the algorithm in [13] for solving $\tilde{\mathbf{P}}$. The adapted version of the algorithm in [13] consists of repeated application of the Pshenichnyi–Pironneau–Polak Min–Max algorithm map to $\tilde{\mathbf{P}}_{N_i}$ until a closed-loop, feedback test determines a switch of stages. We refer to this algorithm as Algorithm RP. Specifically, Algorithm RP switches stages at iteration x_j if the proposed new point z satisfies

$$\psi_{N_i}(z) - \psi_{N_i}(x_j) > -\eta\Delta(N_i)^{\tau}, \tag{33}$$

i.e., z gives too small a cost reduction. Here, $\eta > 0$ and $\tau \in (0, 1)$ are algorithm parameters. The parameter τ is typically set close to one (0.9999 in these numerical examples), while a favorable value of η is difficult to determine. Furthermore, the sample size at each stage must be determined a priori. In this example, we selected five stages with sample sizes equally spaced between the minimum and maximum sample sizes given by the efficient diagonalization strategy, i.e., 50, 443, 836, 1228, and 1621. We used the same random seed in both algorithms. We ran Algorithm RP until $\psi_{1621}(\cdot)$ was equal to the cost achieved in the last iteration of the algorithm with efficient diagonalization. We did not augment the sample size beyond 1621, but continued computing iterates at that stage until the target cost-value was achieved. This is a somewhat favorable stopping criteria for Algorithm RP because this algorithm might augment, prematurely, the sample size beyond 1621 resulting in long computing times. It can be deduced from [13] that Algorithm RP converges almost surely if $\Delta(N) = \sqrt{(\log \log N)/N}$. Hence, we adopted this error function. The computing times for Algorithm RP are summarized in column two of Table 1 for various values of the parameter η in (33). Note that the magnitude of $\psi_N(x)$ is between 10^{-2} and 10^{-3} for feasible x and hence one would expect suitable values of $\eta\Delta(N)^{\tau}$ being somewhat less. In Table 1, the row with $\eta = \infty$ gives the computing time for a fixed sample size equal to the largest sample size 1621 for all iterations.

We also attempted using the error function $\Delta(N) = 1/\sqrt{N}$, which changed the results only marginally as seen in the third column of Table 1. We observe from (33) that a small value of η forces Algorithm RP to solve the current approximating problem accurately before switching to a larger sample size. As seen from Table 1, there is a trade-off between solving the approximating problem accurately at an early stage, potentially wasting time, and solving the early approximations too coarsely, leading to many iteration at stages with high computational cost. If the right balance is found, i.e., a good η, then the feedback rule (33) can be efficient. Of course, it is difficult to select η a priori. To illustrate

Table 1
Computing times [seconds] for Algorithm *RP* [13] applied to Example 1

η	$\Delta(N)$	
	$\sqrt{(\log\log N)/N}$	$1/\sqrt{N}$
∞	980	980
10^{-1}	1044	1036
10^{-2}	1084	654
10^{-3}	678	675
10^{-4}	675	677
10^{-5}	682	676
10^{-6}	476	477
10^{-7}	574	554
10^{-8}	603	601
10^{-9}	898	901

The efficient diagonalization approach computes the same result in 458 s.

this difficulty, we repeated the example for the higher accuracy $\varepsilon = 0.005$. Then, the efficient diagonalization approach increased the sample size up to 6473 and solved the problem in 1461 s. From Table 1 it appears that $\eta = 10^{-6}$ is a good choice. We selected this value and re-solved the problem using Algorithm *RP* with five stages equally spaced in the range [50, 6473] as above. The computing time turned out to be 4729 s. Hence, $\eta = 10^{-6}$ was not efficient in this case.

The second example is a design of a simply supported reinforced concrete *T*-girder for minimum cost according to the specifications in [1], using the design variables $x = (A_s, b, h_f, b_w, h_w, A_v, S_1, S_2, S_3) \in \mathbb{R}^9$, where A_s is the area of the tension steel reinforcement, b is the width of the flange, h_f is the thickness of the flange, b_w is the width of the web, h_w is the height of the web, A_v is the area of the shear reinforcement (twice the cross-section area of a stirrup), and S_1, S_2 and S_3 are the spacings of shear reinforcements in the high, medium, and low shear force zone of the girder, respectively.

We modeled uncertainty using eight independent, random variables collected in a vector V. We assumed that the girder can fail in four different modes corresponding to bending stress in mid-span and shear stress in the high, medium, and low shear force zone. Structural failure occurs if any of the four failure modes occur. This gives rise to four nonlinear, smooth limit-state functions $h_k(x, V)$, $k = 1, 2, 3, 4$, whose exact form is rather complicated and given in [12]. This results in a failure probability $p(x) = P_V[\cup_{k=1}^{4}\{h_k(x, V) \leqslant 0\}]$.

Now, let $\Omega = \{\omega \in \mathbb{R}^8 | \|\omega\| = 1\}$ and P be the uniform distribution on the eight-dimensional hypersphere. Then, similarly to what was done above, the failure probability can be approximated by (30), with

$$G(x, \omega) = \max\left\{\max_{k=1,\ldots,4}\{1 - \chi^2(r_k^2(x, \omega))\}, 1 - \chi^2(a^2)\right\}, \tag{34}$$

$\chi^2(\cdot)$ being the Chi-square cumulative distribution function with 8 degrees of freedom,

$$r_k(x, \omega) = \begin{cases} \min_r\{r | \tilde{h}_k(x, r\omega) \leqslant 0, r \geqslant 0\} & \text{if} \quad \{r | \tilde{h}_k(x, r\omega) \leqslant 0, r \geqslant 0\} \neq \emptyset, \\ \infty & \text{otherwise,} \end{cases} \tag{35}$$

and $\tilde{h}_k(\cdot, \cdot)$ being standardized versions of $h_k(\cdot, \cdot)$. We set the approximation parameter $a = 10$ in this case. We also imposed 24 deterministic, nonlinear constraints as described in [12].

Algorithm parameters were selected to be $n_0 = 50$, $N_0 = 50$, $s = 5$, and $\pi = 1$. Finally, the required fractional reduction in cost $\varepsilon = 0.0001$ and the initial point $x_0^0 = (0.01, 0.5, 0.5, 0.5, 0.5, 0.0005, 0.5, 0.5, 0.5)$ were chosen.

Our proposed algorithm yielded a diagonalization strategy of three stages with 65, 20, and 20 iterations, with sample sizes 50, 373, and 2545, respectively. The total computing time was 1001 s.

Again we compared this result with that obtained using Algorithm *RP* with five stages of equally spaced sample sizes between 50 and 2545. Using the same stopping criterion as for the first example, we obtained the computing times in Table 2.We observe that the computing times using Algorithm *RP* can be significantly longer than those achieved using the efficient diagonalization approach.

Table 2
Computing times [seconds] for Algorithm *RP* [13] applied to Example 2

η	$\Delta(N)$	
	$\sqrt{(\log \log N)/N}$	$1/\sqrt{N}$
∞	>36,000	>36,000
10^{-2}	>12,600	7416
10^{-3}	2004	1990
10^{-4}	2256	2342
10^{-5}	6721	2327
10^{-6}	1209	1608
10^{-7}	11,108	>7200

The efficient diagonalization approach computes the same result in 1001 seconds.

6. Conclusions

We have demonstrated that solving an auxiliary, efficient diagonalization problem to obtain a diagonalization strategy can reduce the overall computing times in stochastic nonlinear programming. In particular, this approach eliminates the need for determining algorithm parameters by means of guesswork or costly numerical experimentation. Instead, the efficient diagonalization problem determines sample sizes and numbers of iterations at each stage using estimated values of cost-to-go, rate of convergence, and sampling error. Even using Matlab, the solution of the diagonalization problem requires only seconds of computing time. Our computational experience indicates that the advantage of an efficient diagonalization approach is more substantial for larger, more complicated problems and when a high-precision solution is sought.

Acknowledgments

The authors are grateful to Professor R.S. Womersley, University of New South Wales, for providing the Matlab-code for solving the efficient diagonalization sub-problem.

References

[1] American Association of State Highway and Transportation Officials, Standard Specifications for Highway Bridges, fifteenth ed., Washington, D.C., 1992
[2] J.V. Burke, Calmness and exact penalization, SIAM J. Control and Optimization 29 (2) (1991) 493–497.
[3] F. Clarke, Optimization and Nonsmooth Analysis, Wiley, New York, NY, 1983.
[4] L. He, E. Polak, Effective diagonalization strategies for the solution of a class of optimal design problems, IEEE Trans. Automat. Control 35 (3) (1990) 258–267.
[5] A.J. King, R.T. Rockafellar, Asymptotic theory for solutions in statistical estimation and stochastic programming, Math. Oper. Res. 18 (1993) 148–162.
[6] A.J. King, R.J.B. Wets, Epi-convergence of convex stochastic programs, Stochastics 34 (1991) 83–91.
[7] M. Ledoux, M. Talagrand, Probability in Banach Spaces, Springer, New York, NY, 1991.
[8] Mathworks, Inc., Natick, Massachusetts, Matlab Reference Manual, Version 7.0, 2004.
[9] E.L. Plambeck, B.R. Fu, S.M. Robinson, R. Suri, Sample-path optimization of convex stochastic performance functions, Math. Programming Ser. B 75 (2) (1996) 137–176.
[10] E. Polak, Optimization Algorithms and Consistent Approximations, Springer, New York, NY, 1997.
[11] E. Polak, J.O. Royset, R.S. Womersley, Algorithms with adaptive smoothing for finite minimax problems, J. Optim. Theory Appl. 119 (3) (2003) 459–484.
[12] J.O. Royset, A. Der Kiureghian, E. Polak, Optimal design with probabilistic objective and constraints, J. Eng. Mech. 132 (1) (2006) 107–118.
[13] J.O. Royset, E. Polak, Extensions of stochastic optimization results to problems with system failure probability functions, J. Optim. Theory and Appl. 132(2) (2007).
[14] A. Ruszczynski, A. Shapiro, Stochastic Programming, Elsevier, New York, NY, 2003.
[15] A. Shapiro, T. Homem-de-Mello, A simulation-based approach to two-stage stochastic programming with recourse, Math. Programming 81 (1998) 301–325.