Retrospective-Approximation Algorithms for the Multidimensional Stochastic Root-Finding Problem

RAGHU PASUPATHY
Virginia Tech
and
BRUCE W. SCHMEISER
Purdue University

The stochastic root-finding problem (SRFP) is that of solving a nonlinear system of equations using only a simulation that provides estimates of the functions at requested points. Equivalently, SRFPs seek locations where an unknown vector function attains a given target using only a simulation capable of providing estimates of the function. SRFPs find application in a wide variety of physical settings.

We develop a family of retrospective-approximation (RA) algorithms called Bounding RA that efficiently solves a certain class of multidimensional SRFPs. During each iteration, Bounding RA generates and solves a sample-path problem by identifying a polytope of stipulated diameter, with an image that bounds the given target to within stipulated tolerance. Across iterations, the stipulations become increasingly stringent, resulting in a sequence of shrinking polytopes that approach the correct solution.

Efficiency results from: (i) the RA structure, (ii) the idea of using bounding polytopes to exploit problem structure, and (iii) careful step-size and direction choice during algorithm evolution. Bounding RA has good finite-time performance that is robust with respect to the location of the initial solution, and algorithm parameter values. Empirical tests suggest that Bounding RA outperforms Simultaneous Perturbation Stochastic Approximation (SPSA), which is arguably the best-known algorithm for solving SRFPs.

Categories and Subject Descriptors: G.1.5 [**Numerical Analysis**]: Roots of Nonlinear Equations—*Iterative methods*; G.3 [**Mathematics of Computing**]: Probability and Statistics—*Probabilistic algorithms (including Monte Carlo)*; I.6.1 [**Simulation and Modeling**]: Simulation Theory

General Terms: Algorithms, Design, Theory

This work was supported in part by the Office of Naval Research through the grant N00014-08-1-0066.

Authors' addresses: R. Pasupathy, Department of Industrial and Systems Engineering, Virginia Tech, Blacksburg VA 2401; email: pasupath@vt.edu; B. W. Schmeiser, Purdue University, West Lafayette, IN 47907.

DOI 10.1145/1502787.1502788 http://doi.acm.org/10.1145/1502787.1502788

Additional Key Words and Phrases: Retrospective approximation, sample-average approximation, stochastic root finding

ACM Reference Format:

Pasupathy, R. and Schmeiser, B. W. 2009. Retrospective-Approximation algorithms for the multidimensional stochastic root-finding problem. ACM Trans. Model. Comput. Simul. 19, 2, Article 5 (March 2009), 36 pages. DOI = 10.1145/1502787.1502788 http://doi.acm.org/10.1145/1502787.1502788

1. INTRODUCTION

Consider the problem of solving a $q \times q$ system of linear, known, and deterministic equations

$$\begin{aligned} g_1(x) = a_{1,1}x_1 + a_{1,2}x_2 + \cdots + a_{1,q}x_q &= \gamma_1 \\ g_2(x) = a_{2,1}x_1 + a_{2,2}x_2 + \cdots + a_{2,q}x_q &= \gamma_2 \\ &\vdots \\ g_q(x) = a_{q,1}x_1 + a_{q,2}x_2 + \cdots + a_{q,q}x_q &= \gamma_q \end{aligned} \tag{1}$$

where $x = (x_1, x_2, \ldots, x_q)^T$. The equations in (1) are *known* because the coefficients, and therefore the functions g_j, $j \in \{1, 2, \ldots, q\}$, are given to us beforehand. The equations are *deterministic* since there is no uncertainty in the given coefficients. The solution to (1) is $x^* = A^{-1}\gamma$, where $\gamma = (\gamma_1, \gamma_2, \ldots, \gamma_q)^T$ and A^{-1} is the inverse of the matrix of coefficients a_{ij}, $i, j \in \{1, 2, \ldots, q\}$.

There is no such solution to the *stochastic root-finding problem* (SRFP), a generalization of (1) where the equations are possibly *nonlinear*, the form of the equations is *unknown* in the sense that function values at particular x values are available only upon request (i.e., the functions are known only through a simulation), and the system is *stochastic* in the sense that the simulation gives only estimates (noisy measurements) of the function values.

1.1 Motivation

SRFPs have recently become ubiquitous. This is partly because they arise routinely as the first-order conditions for optimality in an optimization problem, under conditions where only estimates of the objective function's gradient are available. SRFPs also arise on their own, as the need to solve a system of nonlinear equations with constituent functions that can only be observed. Applications settings are varied and include, amongst others, inventory control, transportation systems, and telecommunication networks. See Chen [1994], Pasupathy [2005], Spall [2003], and Fu [2002] for a long list of motivating settings. Also, most sample problems provided in Ortega and Rheinboldt [1970, Chapter 1] automatically become SRFPs if the constituent functions can only be estimated.

In what follows, we present three applications where SRFPs arise naturally. Real-world applications are diverse, and are usually accompanied by a set of complications that are unique to the context in which they arise. Therefore, any problem statements we provide in the following examples should be appropriately enhanced before implementation.

1.1.1 *Parameter Estimation.* Say a simulation generates output data $\{Y_j\}, Y_j \in \mathbb{R}^q$, that are independent and identically distributed, and known to come from a distribution with the density function $f(y;x^*)$, where x^* is an unknown vector of parameters, and $y \in \mathbb{R}^q$. Suppose we are interested in identifying the specific distribution that generated the data $\{Y_j\}$, that is, we want to identify the vector of parameters x^*.

Since the vector x^* maximizes the function

$$h(x) = \mathrm{E}[\log(f(Y;x))] = \int_{-\infty}^{\infty} \log(f(y;x)) f(y;x^*)\, dy, \tag{2}$$

one way to identify x^* is to solve a maximization problem with $h(x)$ as the objective function. The first-order conditions for this optimization problem are obtained by setting each of the partial derivatives of h to zero, producing the following system of q nonlinear equations. We have

$$h_i(x) = \int_{-\infty}^{\infty} \frac{f_i(y;x)}{f(y;x)} f(y;x^*) dy = 0, \quad i = 1, 2, \ldots, q \tag{3}$$

where $(h_1(x), h_2(x), \ldots, h_q(x))^T$ and $(f_1(x), f_2(x), \ldots, f_q(x))^T$ are the vectors of partial derivatives with respect to x, of h and f ,respectively. Solving (3) is an SRFP because the functions h_i cannot be observed directly but can be estimated as

$$H_i(x;m) = \frac{1}{m} \sum_{j=1}^{m} \frac{f_i(Y_j;x)}{f(Y_j;x)}.$$

1.1.2 *Behavior of Cooperative Retailers in a Supply Chain.* Consider q retailers who form part of a supply chain. Each retailer $i \in \{1, 2, \ldots, q\}$ sells a product that is to be periodically reordered to level x_i. Assume that the retailers cooperate according to prespecified consensual rules of sharing in stock-out situations. For example, each retailer, upon facing inventory shortage, may approach the other retailers (in some order) to satisfy excess demand in return for some percentage of the sale. Let $D_1, D_2, \ldots, D_q$ be random variables (not necessarily independent) representing the demands faced by the retailers $1, 2, \ldots, q$, respectively. Also, let $Y_i, i \in \{1, 2, \ldots, q\}$, be the random variable representing the *optimal* reorder level for retailer i, given the reorder levels $x_j, j \in \{1, 2, \ldots, q\}, j \neq i$, of the other retailers, and given the demands $D_1, D_2, \ldots, D_q$. The problem is to find the equilibrium reorder levels, that is, the reorder levels $x_1, x_2, \ldots, x_q$ such that no retailer can unilaterally change their reorder level and obtain greater profit on average. Formally,

$$\text{find } (x_1, x_2, \ldots, x_q) \text{ such that } (E(Y_1), E(Y_2), \ldots, E(Y_q)) - (x_1, x_2, \ldots, x_q) = 0,$$

given a simulation capable of generating random variates from the multivariate distributions of $(D_1, D_2, \ldots, D_q)$ and $(Y_1, Y_2, \ldots, Y_q)$.

1.1.3 *Determining Toll Prices in Vehicular Traffic Systems.* A method by which traffic is managed in highway networks is toll collection. In a simplistic scenario, a single origin-destination pair in a highway network is connected by q paths on which the toll prices $x_1, x_2, \ldots, x_q$ need to be determined. The

problem is to identify the toll prices $x_1, x_2, \ldots, x_q$ that ensure that the proportions of drivers choosing the various routes match a prespecified set of target proportions on average. Let the random variable $D > 0$ denote the number of drivers who make the trip between the origin-destination pair by choosing one of the q available paths on any day. The drivers choose their paths based on various factors, particularly the toll prices $x_1, x_2, \ldots, x_q$. If the random variables $Y_1, Y_2, \ldots, Y_q$ denote the numbers of drivers choosing the paths $1, 2, \ldots, q$, respectively, the problem of deciding toll prices in vehicular traffic systems is stated as

$$\text{find } (x_1, x_2, \ldots, x_q) \text{ such that } \mathrm{E}\left(\frac{Y_1}{D}, \frac{Y_2}{D}, \ldots, \frac{Y_q}{D}\right) = (\gamma_1, \gamma_2, \ldots, \gamma_q),$$

given $(\gamma_1, \gamma_2, \ldots, \gamma_q)$, and a simulation capable of generating random variates from the distributions of D and $(Y_1, Y_2, \ldots, Y_q)$ as a function of $x_1, x_2, \ldots, x_q$.

1.2 Problem Statement

Formally, the version of SRFP we use in this article is as follows.

Given. (a) a constant vector $\gamma \in \mathbb{R}^q$; and (b) a simulation capable of generating $\overline{Y}_m(x)$ satisfying $\lim_{m\to\infty} \overline{Y}_m(x) \to g(x)$ uniformly (in x), with probability one (w.p.1). Here, m represents some measure of simulation effort.

Find. a root x^* satisfying $g(x^*) = \gamma$ using only the estimator $\overline{Y}_m(x)$.

Algorithms developed for solving SRFPs will be evaluated based on: (i) numerical stability, (ii) robustness, (iii) convergence, (iv) computational efficiency, and (v) the ability to report solution accuracy. See Chen [1994] and Pasupathy [2005] for more on (i), (ii), (iii), (iv), and (v).

To reflect the context in which SRFPs typically occur, we use the term "design space" to refer to the set of possible designs $x \in \mathbb{R}^q$. Similarly, "image space" contains the corresponding responses $\overline{y}_m(x;\underline{\omega}) \in \mathbb{R}^q$, where $\overline{y}_m(x;\underline{\omega})$ denotes a realization of $\overline{Y}_m(x)$ obtained with the random vector $\underline{\omega}$. Throughout the article, we use the phrase "sample-path function" to refer to both $\overline{Y}_m(x)$ and $\overline{y}_m(x;\underline{\omega})$.

1.3 Scope

The SRFP in the original article by Robbins and Monro [1951] was introduced in one dimension, and assumed that the underlying root-finding function g is monotone. Robbins and Monro make no continuity assumptions on g. As we see in the next section, most recent algorithms for solving SRFPs are multidimensional, and typically make some kind of a smoothness assumption on the root-finding function g (e.g., g is thrice differentiable). They also explicitly or implicitly make structural assumptions on the sample-path functions $\{\overline{Y}_m(x)\}$.

The algorithms we present in this article assume that the root-finding function g is *strictly increasing* in multiple dimensions, by which we mean that for $x_1, x_2 \in \mathbb{R}^q, x_1 \neq x_2, (g(x_1) - g(x_2))^T(x_1 - x_2) > 0$ [Ortega and Rheinboldt 1970]. This definition seems like a natural multidimensional extension of the notion of strictly increasing functions, but more importantly, assuming that g is strictly increasing in this sense seems to produce a useful and large (e.g., the gradients of strictly convex functions are strictly increasing as defined) class of SRFPs

which afford the ability to develop tailored algorithms that exploit the underlying structure effectively. One example of this underlying structure is that if $g : \mathbb{R}^q \to \mathbb{R}^q$ is strictly increasing, then the one-dimensional real-valued function formed by projecting g onto any direction d is strictly increasing in one dimension. In other words, if g is strictly increasing in $\mathbb{R}^q$, the function $g_{d,x_0}(t) = g(x_0+td)^T d$ is strictly increasing in $t \in \mathbb{R}$ for any given $x_0, d \in \mathbb{R}^q$. As we see in later sections, we exploit this particular property. We discuss strictly increasing functions in more detail in Section 5.

2. LITERATURE REVIEW

A substantial fraction of the current literature on solving SRFPs can be classified broadly into methods based on stochastic approximation, and those based on sample-path approximation. In the interest of brevity, we limit our discussion of the literature on SRFPs to works falling in these two broad categories. Furthermore, we provide an overview of only the key papers in these categories, and list survey papers whenever available. The enormous recent literature on simulation-optimization (SO) is at least tangentially relevant to solving SRFPs, since SRFPs can be recast as SO problems. See Andradóttir [1998], Fu [2002, 1994], Goldsman and Nelson [1998], Jacobson and Schruben [1989], Kushner and Yin [2003], Safizadeh [1990], Shapiro [1996], and Spall [2003] for extensive reviews on SO and related problems in general.

2.1 Stochastic Approximation

Classical Stochastic Approximation (CSA) is the original stochastic root-finding algorithm developed by Robbins and Monro [1951]. Much of the literature on solving SRFPs is based on CSA. The algorithm as was originally proposed has the simple iterative structure

$$X_{k+1} = X_k - a_k(\overline{Y}_k - \gamma), k = 0, 1, \ldots,$$

where X_0 is the initial guess, $\overline{Y}_k = \sum_{i=1}^{m} Y_i(X_k)/m$, $\{Y_1(x), \ldots, Y_m(x)\}$ is a random sample from the distribution of $Y(x)$, and $\{a_k\}_{k=0}^{\infty}$ is a predetermined sequence of positive constants satisfying $\sum_{i=0}^{\infty} a_k = \infty$ and $\sum_{i=0}^{\infty} a_k^2 < \infty$. Owing to its simple structure, CSA converges in mean square under fairly general conditions. Furthermore, CSA and most of its variants extend naturally to multiple dimensions.

Not surprisingly, CSA does not work well in practice. The algorithm is too inflexible in its direction and step-size choices. The direction choice $\gamma - \overline{Y}_k$ is scale dependent. In other words, the quality of the chosen direction depends critically on the units in which $\overline{Y}_k$ is measured. CSA has been reported to work reasonably well on many SRFPs by appropriately choosing the constants $\{a_k\}$. This implies, however, that the researcher has the option and ability to tune parameters. In situations where the SRFP needs to be solved automatically and rapidly, CSA is not a good choice.

A notable variant of CSA is the Accelerated Stochastic Approximation (ASA) algorithm proposed by Kesten [1958]. ASA tries to address the concern of $\{a_k\}$ being predetermined in CSA by introducing the alternative sequence of

constants $b_0 = a_0$, $b_1 = a_1$, and $b_k = a_{t_k}$ for $k \geq 2$ where

$$t_k = 1 + \sum_{j=2}^{k} I\{(\overline{Y}_{j-1} - \gamma)(\overline{Y}_{j-2} - \gamma) \leq 0\},$$

and

$$I\{A\} = \begin{cases} 1 & \text{if } A \text{ is true;} \\ 0 & \text{otherwise.} \end{cases}$$

As the structure suggests, ASA has a random step-size that decreases only when the previous two iterates have "bracketed" the target value γ. ASA is a slightly improved version of CSA but still suffers from the existence of magic parameters $\{a_k\}$ and the nonutilization of slope information. Following Kesten's work in 1958, various other authors (e.g., Fabian [1968], Venter [1967], Wasan [1969]) have either extended CSA, or expounded on the specific properties of the CSA iterates. More recently, Andradóttir has proposed the Scaled Stochastic Approximation [Andradóttir 1996] and the Projected Stochastic Approximation [Andradóttir 1991] algorithms as modifications to CSA. Scaled Stochastic Approximation, for example, uses the iteration

$$X_{k+1} = X_k - a_k \left(\frac{\overline{Y}_k^{(1)} - \gamma}{\max\{\epsilon, |\overline{Y}_k^{(2)} - \gamma|\}} + \frac{\overline{Y}_k^{(2)} - \gamma}{\max\{\epsilon, |\overline{Y}_k^{(1)} - \gamma|\}} \right), k = 0, 1, \ldots,$$

where ϵ is a predetermined constant, and given X_k, $\overline{Y}_k^{(1)}$ and $\overline{Y}_k^{(2)}$ are two independent estimators of $g(X_k)$ at a specified sample size. These two algorithms, like most other earlier modifications to CSA, focus either on relaxing convergence conditions or on randomizing the step-size sequence while achieving optimal asymptotic efficiency. All these variants of CSA have a simple structure, extend naturally to multiple dimensions, and usually require user-tuning of parameters. See Kushner and Clark [1978], and more recently Kushner and Yin [2003], for a broad coverage of stochastic approximation methods.

Arguably the most popular current method for solving SRFPs is Spall's Simultaneous Perturbation Stochastic Approximation (SPSA) method. The first-order and second-order versions of this method are discussed in Spall [2003, 2000]. The second-order method, called Adaptive Simultaneous Perturbation (ASP), is the true stochastic analog of the modified Newton's method for finding the zeros of a deterministic nonlinear function. One of the important contributions of ASP is the efficient incorporation of gradient estimates for direction finding into the stochastic approximation recursion. Like its deterministic counterpart, and as Spall mentions in Spall [2003, p. 197], ASP has attractive asymptotic properties and performs well when the initial solution is "sufficiently close" to the true root, but generally requires careful choice of algorithm parameters for efficient performance. Automatically choosing parameters in stochastic-approximation-type algorithms to produce good finite-time performance over a wide range of problems has generally been recognized as a challenge [Spall 2000; Yakowitz et al. 2000].

2.2 Sample Average Approximation (SAA)/ Retrospective Approximation (RA)

SAA, also known by various other names including sample-path optimization, the stochastic counterpart method, and retrospective approximation, is another general technique that can be used for solving SRFPs. It seems that the first references to this technique appeared in Healy and Schuben [1991] and Shapiro [1991] in the context of solving SO problems. Several other authors have used the technique in various forms. See, for instance, Atlason et al. [2004], Plambeck et al. [1996], and Rubinstein and Shapiro [1993]. See Homem-de-Mello [2003], Ruszczynski and Shapiro [2003], Shapiro [2000], and Shapiro and Homem-de-Mello [2000] for a thorough discussion of SAA including results on the rates of convergence in the context of SO problems.

The general idea of SAA for solving SRFPs is easily stated. Instead of solving the actual SRFP, solve an approximate problem S obtained by substituting the unknown underlying root-finding function g by the sample-path approximation $\overline{y}_m(x;\underline{\omega}_m)$. The sample-path approximation $\overline{y}_m(x;\underline{\omega}_m)$ is the realization of the consistent estimator $\overline{Y}_m(x)$ of $g(x)$, generated using the vector of random numbers $\underline{\omega}_m = \{\omega_1, \omega_2, \ldots, \omega_m\}$ and sample size m. The sample-path problem S thus has the following simple form.

$$\begin{array}{ll} \text{find } x \text{ such that} & \overline{y}_m(x;\underline{\omega}_m) = \gamma \quad (S) \\ \text{subject to} & x \in \mathbb{R}^q \end{array}$$

So in SAA, instead of the original SRFP, a *single* approximate deterministic problem S generated with a "sufficiently large" sample size is solved using an appropriately chosen deterministic root-finding algorithm. Under certain conditions, most notably on $\overline{y}_m(x;\underline{\omega}_m)$, the solution to S converges to the root x^* of g as the sample size $m \to \infty$.

This article deals with Retrospective Approximation (RA), a refinement of SAA where instead of generating and solving a *single* sample-path problem S, a *sequence* of sample-path problems $\{S_k\}$ are generated with increasing sample sizes $\{m_k\} \to \infty$, and solved to decreasing tolerances $\{(\epsilon_k, \eta_k)\}$, where ϵ_k and η_k are specified tolerances in the design space and image space, respectively. RA was originally proposed by Chen and Schmeiser [2001] to solve one-dimensional SRFPs. The philosophy behind the RA structure is as follows: During the early iterations, use small sample sizes m_k and large error tolerances (ϵ_k, η_k) in solving the sample-path problem S_k; in later iterations, as the root estimate $\overline{X}_k$ tends closer to the true root x^*, use larger sample sizes and correspondingly smaller error tolerances. The early iterations are efficient because the small sample sizes ensure that not much computing effort is expended in generating sample-path problems, and the later iterations are efficient because the root estimate $\overline{X}_k$ is probably close to the true root x^* and not much effort is expended in solving sample-path problems. This structure of RA is central to the simultaneous goals of proving convergence and achieving good practical performance.

3. BOUNDING RA ALGORITHMS

In this section we present the Bounding RA family of algorithms to solve multidimensional SRFPs. We first formally present the generic RA framework.

RA Components:

(i) a procedure for solving the sample-path equation (S_k) to specified tolerance vector (ϵ_k, η_k);
(ii) a rule to compute the sample size sequence $\{m_k\}$; and
(iii) a rule to compute the error tolerance sequence $\{(\epsilon_k, \eta_k)\}$.

RA Logic:

(0) Initialize the retrospective iteration number $k = 1$. Set m_1, ϵ_1, η_1.
(1) Generate $\underline{\omega}_k$.
(2) Use RA component (i) to solve the deterministic sample-path equation (S_k) to within (ϵ_k, η_k). Obtain a retrospective solution X_k.
(3) Calculate the root estimate $\overline{X}_k$ as the weighted sum of retrospective solutions $\{X_i\}_{i=1}^k$.
(4) Terminate if desired precision is attained (Section 7.3). Otherwise, use RA components (ii) and (iii) to get $m_{k+1}, \epsilon_{k+1}, \eta_{k+1}$. Set $k \leftarrow k + 1$ and go to 1.

Specific RA algorithms result from making choices for the three RA components. For instance, Bounding RA is a subfamily of RA algorithms characterized by their choice of component (i) in RA. Bounding RA algorithms use the Bounding Algorithm (BA), to be presented, to solve (S_k) by finding a design polytope P_k of diameter less than design tolerance ϵ_k, and with an image polytope that bounds the target γ to within response tolerance η_k.

We shortly clarify (Section 3.1) the notions of bounding, design polytope, image polytope, design tolerance, and image tolerance. For now, we emphasize that the broad idea of Bounding RA algorithms is identifying a sequence $\{P_k\}$ of shrinking polytopes in the design space whose images decrease in distance from the target γ. The kth retrospective solution X_k is set to some point within the polytope P_k identified during the kth iteration.

3.1 Definition of Bounding

In what follows, when we refer to a polytope $yP = \{y_1, y_2, \ldots, y_n\}$, we are referring to the closed convex hull of the points $y_1, y_2, \ldots, y_n$. We sometimes refer to a *design polytope* $P \equiv \{Z_0, Z_1, \ldots, Z_n\}$, by which we mean that P is the closed convex hull of the points $Z_0, Z_1, \ldots, Z_n$ located in the *design space*. The image or the image polytope yP of the design polytope $P \equiv \{Z_0, Z_1, \ldots, Z_n\}$ is the closed convex hull of the points $\overline{y}_{m_k}(Z_0; \underline{\omega}_k), \overline{y}_{m_k}(Z_1; \underline{\omega}_k), \ldots, \overline{y}_{m_k}(Z_n; \underline{\omega}_k)$ lying in the image space. A design polytope and its image are illustrated in Figure 1. Also, the diameter $v(P)$ of a polytope P has its usual meaning $v(P) = \sup_{z,z'}\{\|z - z'\| : z, z' \in P\}$. Here, the supremum is attained because the polytope P is closed.

Successful termination of the kth iteration in Bounding RA algorithms results in the identification of a design polytope P_k, of diameter less than design tolerance ϵ_k, with an image that "bounds the target γ to within response tolerance η_k." We now make the notion of *bounding* precise and discuss reasons for our choice. For ease of exposition of bounding, we first introduce the notion of *straddling*.

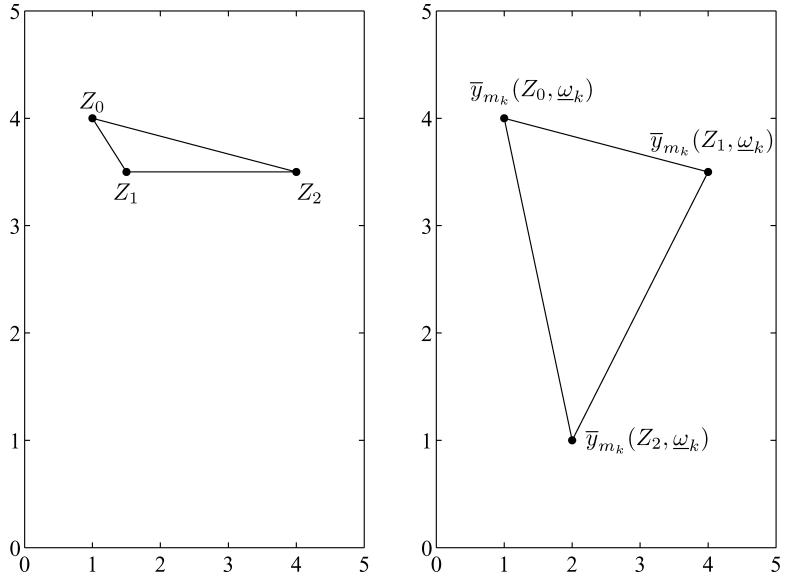

Fig. 1. Illustration of a design polytope and its image polytope. The closed convex hull of the points Z_0, Z_1, Z_2 in the design space is the design polytope. The closed convex hull of the points $\overline{y}_{m_k}(Z_0; \underline{\omega}_k), \overline{y}_{m_k}(Z_1; \underline{\omega}_k), \overline{y}_{m_k}(Z_2; \underline{\omega}_k)$ in the image space is the image polytope.

Definition 3.1. The pair of points $y_1, y_2 \in \mathbb{R}^q$ *straddle* $\gamma \in \mathbb{R}^q$ along the vector $d \in \mathbb{R}^q$ if the interval formed by $y_1^T u_d$, $y_2^T u_d$ contains $\gamma^T u_d$, namely, if

$$\gamma^T u_d \in [\text{Min}(y_1^T u_d, y_2^T u_d), \text{Max}(y_1^T u_d, y_2^T u_d)],$$

where $u_d = d/\|d\|$ is the unit vector parallel to d.

Stated differently, let y_{1p}, y_{2p}, and γ_p be the projections of points y_1, y_2, and γ onto a line L parallel to vector d. Then the points y_1, y_2 *straddle* γ along the vector d if γ_p lies in the *line segment* $L_{y_{1p}, y_{2p}}$ formed by joining y_{1p} and y_{2p}. This is illustrated in Figure 2, where pairs of points such as y_1, y_2, or y_1, y_3 straddle γ along d, but pairs such as y_3, y_4 or y_2, y_4 do not.

Similarly, the two points $y_1, y_2 \in \mathbb{R}^q$ are said to *straddle* γ along the vector $d \in \mathbb{R}^q$ *to within* $\eta > 0$ if the largest interval formed by the points $y_1^T u_d \pm \eta$, $y_2^T u_d \pm \eta$ contains $\gamma^T u_d$, namely if

$$\gamma^T u_d \in [\text{Min}(y_1^T u_d, y_2^T u_d) - \eta, \text{Max}(y_1^T u_d, y_2^T u_d) + \eta].$$

We see from the previous definition that two points which straddle γ along the vector d also straddle γ along the vector $-d$. Also, larger values of η facilitate straddling.

We are now ready to define the notion of bounding.

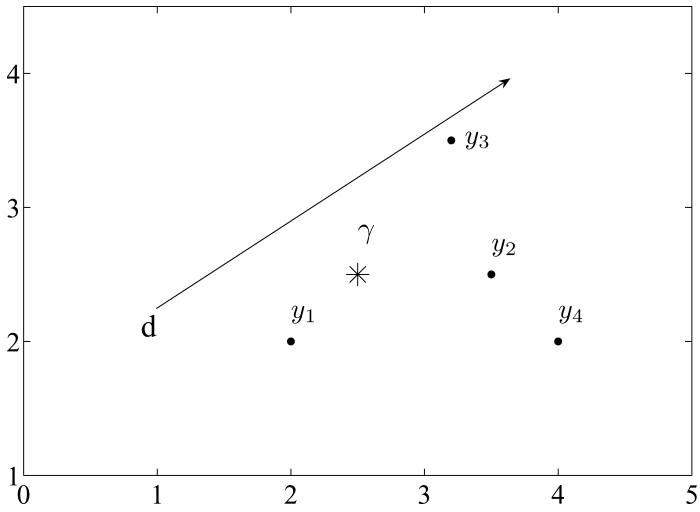

Fig. 2. Illustration of straddling along a vector. The points y_1, y_2 straddle the target γ along the vector d. The points y_3, y_4 do not.

Definition 3.2. The polytope $yP \equiv \{y_0, y_1, \ldots, y_n\}$ is said to *bound* the target γ with respect to the orthogonal set $\{d_1, d_2, \ldots, d_q\}$ if for every $j \in \{1, 2, \ldots, q\}$ there exist $y_{j_1}, y_{j_2} \in \{y_0, y_1, \ldots, y_q\}$ that straddle γ along the vector d_j.
Likewise, the polytope $yP \equiv \{y_0, y_1, \ldots, y_n\}$ *bounds* the target γ with respect to the orthogonal set $\{d_1, d_2, \ldots, d_q\}$ to within $\eta_k \in \mathbb{R}^+$ (the set of positive real numbers) if for every $j \in \{1, 2, \ldots, q\}$ we can find $y_{j_1}, y_{j_2} \in \{y_0, y_1, \ldots, y_q\}$ that straddle γ along the vector d_j to within η_k.

The idea of bounding is illustrated in Figure 3 for $\eta = 0$, where the polytope yP_1 formed as the convex hull of y_1, y_2, y_3 bounds γ with respect to $\{d_1, d_2\}$ since y_1, y_2 straddle γ along d_1, and y_1, y_3 straddle γ along d_2. By contrast, there exists no pair of points among y_4, y_5, y_6, y_7 that straddle γ along the vector d_1. Thus, the polytope yP_2 formed as the convex hull of y_4, y_5, y_6, y_7 does not bound γ with respect to $\{d_1, d_2\}$.

When discussing a polytope bounding the target γ with respect to the orthogonal set $\{d_1, d_2, \ldots, d_q\}$ to within η_k, we often omit the phrase "to within tolerance η_k" if $\eta_k = 0$. Also, we omit the phrase "with respect to the orthogonal set $\{d_1, d_2, \ldots, d_q\}$" when the orthogonal set being referred to is evident or unimportant.

There are two key reasons for our somewhat elaborate definition of bounding. First, bounding as defined in Definition 3.2, is a *computationally efficient* measure of the proximity of a given polytope yP_k to a given target γ. As we shall see, with bounding defined as before, the check for bounding is particularly easy when used within the context of BA, where the design polytope is constructed gradually as BA evolves. By contrast, the alternative definition $\gamma \in yP_k$ is more computationally expensive, requiring the solution of a quadratic programming problem each time a bounding check needs to be performed.

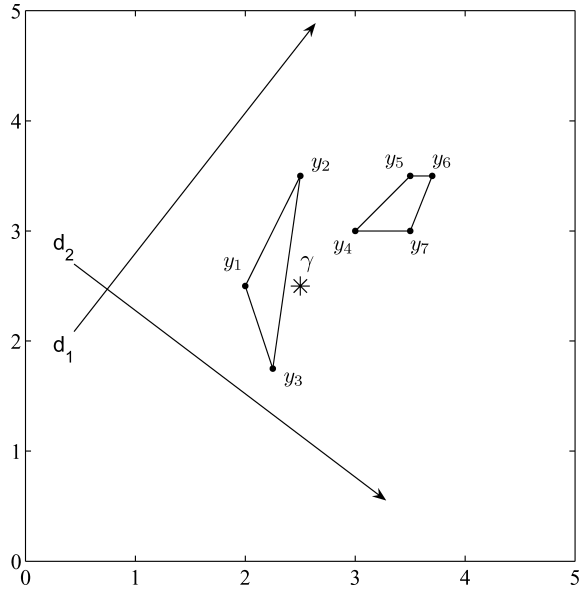

Fig. 3. Illustration of bounding. Unlike the polytope formed by the points y_4, y_5, y_6, y_7, the polytope formed by the points y_1, y_2, y_3 bounds the target γ with respect to the orthogonal set $\{d_1, d_2\}$. This is because y_1, y_2 straddle the target γ along vector d_1, and y_1, y_3 straddle the target γ along vector d_2.

Furthermore, bounding in Definition 3.2 is a relaxation of the condition $\gamma \in y P_k$ since bounding essentially means that γ belongs to the neighborhood of the polytope $y P_k$.

Second, we have defined bounding in a way that facilitates the successful termination of the kth iteration even when the sample-path function does not actually attain the target γ, that is, even when the sample-path problem S_k has no solution. The sample-path function $\overline{y}_{m_k}(x; \underline{\omega}_k)$ not attaining the target γ is a distinct possibility, especially when they are step functions estimating a probabilistic performance measure. This is best illustrated in one dimension. Let g be a cumulative distribution function (cdf), and let $\overline{y}_{m_k}(x; \underline{\omega}_k)$ be an empirical cdf (step function) constructed from m_k observations. Also let the target $\gamma = 0.5$, but say $\overline{y}_{m_k}(x; \underline{\omega}_k)$ does not actually attain it. Under Definition 3.2, irrespective of how small the stipulated tolerance ϵ_k is, successful termination can be achieved since two arbitrarily close points x_1, x_2 can be found so that $\overline{y}_{m_k}(x_1; \underline{\omega}_k) \leq \gamma$ and $\overline{y}_{m_k}(x_2; \underline{\omega}_k) \geq \gamma$. In this case, the line segment joining x_1 and x_2 is the design polytope with diameter $|x_2 - x_1|$, and the line segment joining $\overline{y}_{m_k}(x_1; \underline{\omega}_k)$, $\overline{y}_{m_k}(x_2; \underline{\omega}_k)$ is the image polytope. By contrast, successful termination is unattainable with a termination criterion such as "identify an x such that $|\overline{y}_{m_k}(x; \underline{\omega}_k) - \gamma| \leq \eta_k$," if η_k is chosen too small.

We note in passing that Lemma 3.3 provides an upper bound on the nearest and farthest distances between the target γ and its bounding polytope $y P_k$. The proof of Lemma 3.3 is provided in the online Appendix in the ACM Digital Library.

LEMMA 3.3. *If the polytope* $y P_k \equiv \{y_1, y_2, \ldots, y_n\}$ *bounds* γ *with respect to the orthogonal set* $\{d_1, d_2, \ldots, d_q\}$ *to within* η_k*, then*

(i) $\inf\{\|y - \gamma\| : y \in y P_k\} \leq \sqrt{q}(v(y P_k) + \eta_k)$;
(ii) $\sup\{\|y - \gamma\| : y \in y P_k\} \leq (\sqrt{q} + 1)(v(y P_k) + \eta_k)$,

where $v(y P_k)$ *is the diameter of the polytope* $y P_k$.

3.2 Sample-Path Problem Statement

The sample-path problem is that of finding a small enough design polytope whose image polytope bounds the target γ to within a specified tolerance. Formally, the sample-path problem statement is as follows.

Given. The initial solution $\overline{X}_{k-1} \in \mathbb{R}^q$, the target $\gamma \in \mathbb{R}^q$, tolerances $(\epsilon_k, \eta_k) \in \mathbb{R}^+ \times \mathbb{R}^+$, and a simulation that reveals for each $x \in \mathbb{R}^q$, and a vector of random numbers $\underline{\omega}_k$, a value $\overline{y}_{m_k}(x; \underline{\omega}_k) \in \mathbb{R}^q$.

Find. A design polytope P_k of diameter $v(P_k) \leq \epsilon_k$, and whose image polytope $y P_k$ bounds the target γ to within η_k.

3.3 Solving the Sample-Path Problem

We now present the Bounding Algorithm (BA), which is the deterministic root-finding logic used by Bounding RA algorithms in solving the sample-path problem (S_k). As stipulated by the iteration-level problem statement, termination of BA implies the identification of a small enough design polytope whose image polytope bounds the target γ.

3.3.1 *Operations in BA.* We now define some terminology and present the three operations in BA.

Definition 3.4. The search is based on the current design line segment defined by an old end and a new end, both in the design space. The image line segment is the line joining the images of the old and new ends of the current design line segment.

Design line segments form the search mechanism for BA. During the kth iteration, BA constructs the required design polytope P_k using $n, 1 \leq n \leq q$, orthogonal design line segments, each having a common old end Z_0, and new ends $Z_1, Z_2, \ldots, Z_n$. Each of the resulting pairs of images $\{\overline{y}_{m_k}(Z_0; \underline{\omega}_k), \overline{y}_{m_k}(Z_i; \underline{\omega}_k)\}, i = 1, 2, \ldots, n$, straddle the target γ along $Z_i - Z_0$ to within η_k. This ensures that the image polytope $y P_k$ formed as the convex hull of $\overline{y}_{m_k}(Z_0; \underline{\omega}_k)$, $\overline{y}_{m_k}(Z_1; \underline{\omega}_k)$, $\overline{y}_{m_k}(Z_2; \underline{\omega}_k)$, …, $\overline{y}_{m_k}(Z_n; \underline{\omega}_k)$ bounds γ with respect to the orthogonal set $\{Z_1 - Z_0, Z_2 - Z_0, \ldots, Z_n - Z_0\}$ to within η_k. This is illustrated in Figure 4 for $n = q = 2$ and $\eta_k = 0$.

We now present the three operations used in constructing the bounding polytope $y P_k$. For ease of exposition, we assume that $\eta_k = 0$.

(1) *Grafting* is searching the design space until a design line segment with old end Z_i and new end Z_j is identified so that the points $\overline{y}_{m_k}(Z_i; \underline{\omega}_k)$, $\overline{y}_{m_k}(Z_j; \underline{\omega}_k)$ straddle γ along the vector $Z_j - Z_i$. So, in Figure 5(a), grafting results in the identification of the design line segment L_{Z_2, Z_3} whose image

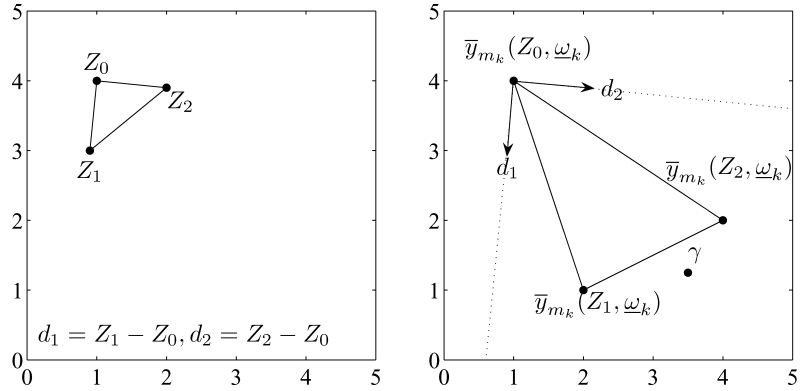

Fig. 4. BA constructs the required design polytope using orthogonal design line segments, each having a common old end Z_0. So, in Figure 4, L_{Z_0,Z_1} and L_{Z_0,Z_2} are the design line segments, and the closed convex hull of Z_0, Z_1, Z_2 is the design polytope. The image polytope bounds the target γ with respect to $\{d_1, d_2\}$ since $\overline{y}_{m_k}(Z_0;\underline{\omega}_k)$, $\overline{y}_{m_k}(Z_1;\underline{\omega}_k)$ straddle γ along d_1, and $\overline{y}_{m_k}(Z_0;\underline{\omega}_k)$, $\overline{y}_{m_k}(Z_2;\underline{\omega}_k)$ straddle γ along d_2.

has ends $\overline{y}_{m_k}(Z_2;\underline{\omega}_k)$, $\overline{y}_{m_k}(Z_3;\underline{\omega}_k)$ straddling the target γ along the vector $Z_3 - Z_2$. If the function $\overline{y}_{m_k}(x;\underline{\omega}_k)$ "looks like" the limit function g, then larger design line segments facilitate straddling. So, in order to increase chances of straddling, the searching during the grafting phase is done with design line segments of *progressively increasing size*.

(2) *Pruning* is whittling the design line segment obtained after grafting to the stipulated size ϵ_k if necessary. Whittling is done while ensuring that the ends of the resulting image line segment continue to straddle the target γ. In Figure 5(b), the design line segment L_{Z_2,Z_3} is whittled and relabeled to L_{Z_0,Z_1}. The points $\overline{y}_{m_k}(Z_0;\underline{\omega}_k)$, $\overline{y}_{m_k}(Z_1;\underline{\omega}_k)$ straddle the target γ along the vector $Z_1 - Z_0$. After the pruning step the design polytope is always the design line segment L_{Z_0,Z_1}.

(3) *Growing* is expanding the design polytope to increase chances of termination through bounding. Growing succeeds pruning and is aimed at constructing the next among the n, $1 \le n \le q$, orthogonal design line segments with ends having images that straddle the target γ. In Figure 5(c), the design polytope L_{Z_0,Z_1} obtained after pruning is augmented with an orthogonal design line segment L_{Z_0,Z_2} of size ϵ_k. The closed convex hull of Z_0, Z_1, Z_2 is the new design polytope, and its image polytope is the closed convex hull of the points $\overline{y}_{m_k}(Z_0;\underline{\omega}_k)$,$\overline{y}_{m_k}(Z_1;\underline{\omega}_k)$,$\overline{y}_{m_k}(Z_2;\underline{\omega}_k)$.

3.3.2 *BA Logic and Listing.* BA, at all times during evolution, maintains a *current* design line segment with an old end and a new end, and a *current* design polytope. At the beginning of the kth iteration, the old end and the new end are set to the root estimate $\overline{X}_{k-1}$. The current design polytope P_k is initialized to $\{\overline{X}_{k-1}\}$. BA then evolves by repeating the following three steps in sequence until a design polytope with an image polytope that bounds the target is found. Each time the current design polytope is updated, the bounding termination criterion is checked.

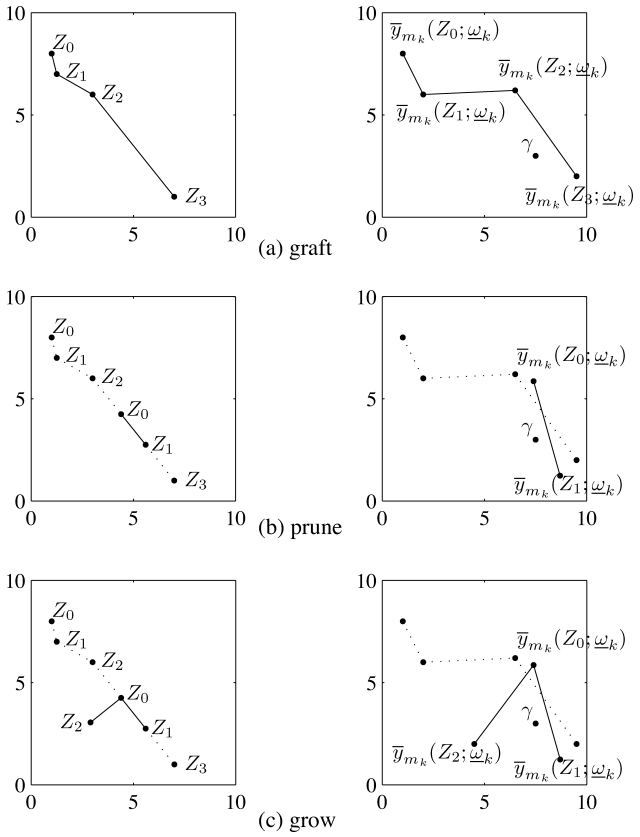

Fig. 5. BA consists of three operations: (a) graft, (b) prune, and (c) grow. In Figure 5(a), two points Z_2, Z_3 are found so that their images $\overline{y}_{m_k}(Z_2;\underline{\omega}_k)$, $\overline{y}_{m_k}(Z_3;\underline{\omega}_k)$ straddle the target γ along $Z_3 - Z_2$. In Figure 5(b), L_{Z_2,Z_3} is pruned to L_{Z_0,Z_1} while ensuring that $\overline{y}_{m_k}(Z_0;\underline{\omega}_k)$, $\overline{y}_{m_k}(Z_1;\underline{\omega}_k)$ still straddle the target γ along $Z_3 - Z_2$. In Figure 5(c), the current design polytope L_{Z_0,Z_1} is grown by augmenting a new design line segment L_{Z_0,Z_2}.

—*(Graft and Update)*. Search the design space (e.g., use the Newton direction with gradient estimated via the Broyden update [Broyden 1965]) to find a design line segment with old end Z_i and new end Z_j such that the points $\overline{y}_{m_k}(Z_i;\underline{\omega}_k)$, $\overline{y}_{m_k}(Z_j;\underline{\omega}_k)$ straddle the target γ along the vector $Z_j - Z_i$. Set the current design polytope and the current design line segment to L_{Z_i,Z_j}.

—*(Prune and Update)*. If the diameter of L_{Z_i,Z_j} is larger than the error tolerance ϵ_k, then prune (e.g., using a bisection search) the design line segment to required size to obtain the design line segment L_{Z_0,Z_1}. Set the current design line segment and the current design polytope to L_{Z_0,Z_1}.

—*(Grow and Update)*. Find a point Z_j so that $\|Z_j - Z_0\| = \epsilon_k$ and $Z_j - Z_0$ is orthogonal to the current design polytope. Set the current design line segment to L_{Z_0,Z_j}, and update the current design polytope to the closed convex hull of the current design polytope and the point Z_j. If the points $\overline{y}_{m_k}(Z_0;\underline{\omega}_k)$, $\overline{y}_{m_k}(Z_j;\underline{\omega}_k)$ straddle the target γ along the vector $Z_j - Z_0$ then

the growing step is repeated. Otherwise, the current design line segment and the current design polytope are set to L_{Z_0,Z_j}, and we go back to the grafting step.

3.3.3 *BA Listing*. *Given*. Initial guess $\overline{X}_{k-1}$, target γ, tolerances $(\epsilon_k, \eta_k) \in \mathbb{R}^+ \times \mathbb{R}^+$, sample size m_k, and a simulation that returns $\overline{y}_{m_k}(x;\underline{\omega}_k)$ for each $x \in \mathbb{R}^q$ and a vector of random numbers $\underline{\omega}_k$.

Find. Design polytope P_k such that: (i) $v(P_k) \leq \epsilon_k$, and (ii) yP_k bounds γ to within η_k.

(0) Set $Z_0 = \overline{X}_{k-1}$.
(1) Simulate at Z_0 to obtain $\overline{y}_{m_k}(Z_0;\underline{\omega}_k)$.
(2) Initialize $Z_1 = Z_0$, $n = 1$, $\delta = \epsilon_k$, $P_k = \{Z_0\}$, $d = (\gamma - \overline{y}_{m_k}(Z_0;\underline{\omega}_k))/\|\gamma - \overline{y}_{m_k}(Z_0;\underline{\omega}_k)\|$.
(3) Graft and update: While $\overline{y}_{m_k}(Z_{n-1};\underline{\omega}_k)^T d - \gamma^T d < 0$ and $\overline{y}_{m_k}(Z_n;\underline{\omega}_k)^T d - \gamma^T d < -\eta_k$, repeat steps 3(a)–(e).
  (a) Generate a unit direction vector d such that $\overline{y}_{m_k}(Z_n;\underline{\omega}_k)^T d - \gamma^T d < 0$.
  (b) Set $n = n + 1$, $Z_n = Z_{n-1} + \delta d$.
  (c) Simulate at Z_n to obtain $\overline{y}_{m_k}(Z_n;\underline{\omega}_k)$.
  (d) Set step size: $\delta = c_2\delta$.
  (e) Update $P_k \equiv \{Z_{n-1}, Z_n\}$.
(4) Initialize $Z_0 = Z_{n-1}$, $Z_1 = Z_n$.
(5) Prune and update: While $\|Z_1 - Z_0\| > \epsilon_k$ repeat steps 5(a)–(d).
  (a) Set $Z = (Z_0 + Z_1)/2$.
  (b) Simulate at Z to obtain $\overline{y}_{m_k}(Z;\underline{\omega}_k)$.
  (c) If $\overline{y}_{m_k}(Z;\underline{\omega}_k)^T d - \gamma^T d < -\eta_k$, then set $Z_0 = Z$. Otherwise set $Z_1 = Z$.
  (d) Update $P_k \equiv \{Z_0, Z_1\}$.
(6) Initialize $O = \{(Z_1 - Z_0)/\|Z_1 - Z_0\|\}$.
(7) Grow and update: Repeat steps 7(a)–(d) at most $q - 1$ times.
  (a) Generate unit vector d orthogonal to O such that $\overline{y}_{m_k}(Z_0;\underline{\omega}_k)^T d - \gamma^T d < 0$.
  (b) Set $O = \{O, d\}$.
  (c) Simulate at $Z_0 + d\epsilon_k$ to obtain $\overline{y}_{m_k}(Z_0 + d\epsilon_k;\underline{\omega}_k)$.
  (d) If $\overline{y}_{m_k}(Z_0 + d\epsilon_k;\underline{\omega}_k)^T d - \gamma^T d < -\eta_k$, set $Z_0 = Z_0 + d\epsilon_k$ and go to step 3. Otherwise, update $P_k = \{P_k, Z_0 + d\epsilon_k\}$.
(8) Return P_k.

The bounding condition is checked each time the polytope P_k is updated.

4. CONVERGENCE OF BOUNDING RA ALGORITHMS

In this section, we discuss the convergence of the sequence of retrospective solutions in Bounding RA. To facilitate exposition, we first discuss and present a broad outline of the convergence proof. The various results comprising proof of convergence are presented in Section 4.2. Section 5 provides intuition on the assumptions made in arriving at the convergence results, specifically on *increasing* and *conservative* function classes.

4.1 Outline of Convergence

Recall that the Bounding RA algorithm for solving SRFPs has the following form.

(0) Initialize the retrospective iteration number $k = 1$. Set m_1, ϵ_1, η_1.
(1) Generate $\underline{\omega}_k$.
(2) Use the BA algorithm to solve the deterministic sample-path equation (S_k) to within (ϵ_k, η_k). Obtain a retrospective solution X_k.
(3) Calculate the root estimate $\overline{X}_k$ as the weighted sum of retrospective solutions $\{X_i\}_{i=1}^k$.
(4) Estimate the precision of the root estimate $\overline{X}_k$, set $m_{k+1}, \epsilon_{k+1}, \eta_{k+1}$, set $k \leftarrow k + 1$, and go to 1.

The Bounding RA algorithm, as listed, has the form of an "outer stochastic" part that embeds an "inner deterministic" part. The inner deterministic part comprises only step 2, for which we use the BA algorithm described in Section 3.3. We establish convergence through corresponding results on the inner and outer parts of Bounding RA. Specifically, we establish convergence through the following results.

—Theorem 4.2, relying on Lemma 4.1, proves that the BA algorithm in step 2 (inner deterministic part) of Bounding RA always terminates successfully, if certain weak conditions hold. Recall that the successful termination of the BA algorithm results in the identification of a design polytope P_k whose image polytope bounds the target γ. Any point in P_k can be chosen as the retrospective solution X_k.
—Theorem 4.4 shows that if step 2 in the Bounding RA algorithm indeed terminates successfully for all k as Theorem 4.2 asserts, then for arbitrarily small $\beta > 0$, $\|g(X_k) - \gamma\| \leq \beta$ for large enough k w.p.1, where γ is the target, and x^* is the true root satisfying $g(x^*) = \gamma$. Equivalently, for arbitrarily small $\beta > 0$, $\|X_k - x^*\| \leq \beta$ for large enough k w.p.1.

We see from the preceding outline that the convergence we establish is weaker than the usual notion of convergence *to a desired point*. Instead we only show that the generated sequence of retrospective solutions $\{X_k\}$, and their images $\{g(X_k)\}$, converge to an arbitrarily small *neighborhood* of the true root x^*, and the target γ, respectively.

We present all our convergence results in terms of the sequence of retrospective solutions $\{X_k\}$, as opposed to the sequence of root estimates $\{\overline{X}_k\}$. (Recall that the root estimate $\overline{X}_k$ is obtained as a weighted combination of the retrospective solutions $X_1, X_2, \ldots, X_k$.) This is purely from the standpoint of mathematical convenience, and the convergence proofs that we present extend seamlessly to the sequence $\{\overline{X}_k\}$ under mild conditions on the weights used to calculate the root estimates. During actual implementation, however, we suggest reporting the root estimates $\{\overline{X}_k\}$ because they have superior statistical properties. Specifically, when the sample sizes are used as weights, namely if $\overline{X}_k = \sum_{i=1}^k m_i X_i / \sum_{i=1}^k m_i$, and the retrospective solutions

$X_1, X_2, \ldots, X_k$ are unbiased estimators of x^* with variance having the canonical form $\mathrm{Var}(X_i) = \sigma^2/m_i$, the root estimate $\overline{X}_k$ happens to be the best (least variance) unbiased estimator.

4.2 Convergence

In this section, we present the various results outlined in Section 4.1. The following two assumptions, stated in generic form, are used to varying extents in arriving at the results. These assumptions, particularly their stringency, are discussed in Section 5.

(A1) A function $f : \mathbb{R}^q \to \mathbb{R}^q$ is *increasing* if for $x_1, x_2 \in \mathbb{R}^q$, $(f(x_1) - f(x_2))^T(x_1 - x_2) \geq 0$; It is *strictly increasing* if the inequality in the definition of increasing functions holds strictly when $x_1 \neq x_2$.

(A2) A function $f : \mathbb{R}^q \to \mathbb{R}^q$ is *conservative* if the line integral of f around any closed loop is zero, namely $\oint f^T \mathbf{dr} = 0$.

We note that strictly increasing functions are one-to-one functions, thus implying that the true root x^*, if present, has to be unique. We rely on this property for proving convergence, and formally state and prove it in the Appendix (Theorem A1).

We now introduce some notation for ease of exposition. In solving the kth sample-path problem (S_k), BA might perform the grafting operation a few times. Each time this operation is performed, a number of points $Z_0, Z_1, \ldots, Z_n$ are visited. Recall that in this case, $\overline{y}_{m_k}(Z_n; \underline{\omega}_k)$ and $\overline{y}_{m_k}(Z_{n-1}; \underline{\omega}_k)$ straddle the target γ along $Z_n - Z_{n-1}$. For the purposes of this section, we take the set of points $Z_0, Z_1, \ldots, Z_n$ visited during each grafting operation, omit the last element Z_n of each of these sets, and form a new set $\mathbb{G}_k$ through augmentation and relabeling. For example, during the kth iteration, say BA performs the grafting operation twice and visited the points Z_0, Z_1, Z_2, Z_3 during the first operation, and the points Z_0, Z_1, Z_2 during the second operation. We form a new set $\mathbb{G}_k = \{Z_0, Z_1, Z_2, Z_3, Z_4\}$, where the points Z_0, Z_1, Z_2 are the first three points visited during the first grafting operation, and the points Z_3, Z_4 are the first two points (relabeled) visited during the second grafting operation. The *trajectory* of BA during the kth iteration is then the piecewise-linear curve that passes through the points in the set $\mathbb{G}_k$, in order. So, the trajectory starts at $Z_0 = \overline{X}_{k-1}$, connects points $Z_1, Z_2, \ldots$ with line segments, and ends at one of the vertices of the design polytope P_k that is returned upon termination. Denote this trajectory, *parameterized by length* s, as $r_k(s), 0 \leq s \leq l_k$, where l_k is the total length of the trajectory. Therefore, the point $r_k(s)$ refers to the point on the trajectory that is s units away from Z_0 when measured along the trajectory.

Lemma 4.1 states that the trajectory of BA remains finite for every k w.p.1, under certain conditions on the sample-path functions $\{\overline{y}_{m_k}(x; \underline{\omega}_k)\}$. We emphasize that the result only asserts that the trajectory, during any iteration, does not head to infinity, namely, that an infinite amount of time is not spent on any one iteration. The length of the trajectory can depend on the iteration number and the specific realization.

Before we formally state the results, in order to avoid confusion, it may be useful to clarify the probability space within which almost sure convergence statements are made. Two levels of probability spaces are implicit in our results: an iteration-level outcome space Ω_k formed as the m_k-fold crossproduct of an underlying implicit probability space (formed as a result of sampling), and an outer "giant" probability space Ω formed as the infinite crossproduct of the probability spaces Ω_k. An outcome $\omega \in \Omega$ thus corresponds to a sequence of iteration-level outcomes $\{\underline{\omega}_k\}$, $\underline{\omega}_k \in \Omega_k$ and a corresponding sequence of sample-path functions $\{\overline{y}_{m_k}(x;\underline{\omega}_k)\}$. All statements of almost sure convergence are made with respect to the space Ω.

LEMMA 4.1. *Suppose that, for each k, the sample-path function $\overline{y}_{m_k}(x;\underline{\omega}_k)$ is increasing and conservative, and there exists X_k^* satisfying $(\overline{y}_{m_k}(x;\underline{\omega}_k)-\gamma)^T(x-X_k^*) > 0$ for all x in some neighborhood of X_k^*, w.p.1. Then, for given positive-valued tolerances ϵ_k and η_k, the trajectory $r_k(s)$, $0 \le s \le l_k$ of BA during the kth iteration of Bounding RA algorithms is finite for all k w.p.1; In other word, for each k, there exists $M(k) > 0$ such that $\|r_k(s_1) - r_k(s_2)\| < M(k)$ for any s_1, s_2 satisfying $0 \le s_1 \le s_2 \le l_k$, w.p.1.*

PROOF. See online Appendix. □

The assumption $(\overline{y}_{m_k}(x;\underline{\omega}_k) - \gamma)^T(x - X_k^*) > 0$ is a multidimensional generalization of the "crossing root" idea used in Chen and Schmeiser [2001], and an analog of similar assumptions in Blum [1954]. Intuitively, this stipulation only means that there exists a point X_k^* which separates the design space into regions whose images "lie on either side of the target γ." This stipulation does not imply that the sample-path function actually attains the target γ, and is intended only to exclude pathological sample-path functions which make the BA algorithm cycle indefinitely.

Theorem 4.2 uses Lemma 4.1 to prove that the sample-path problem (S_k) is solved successfully for every k w.p.1.

THEOREM 4.2. *Let the assumptions of Lemma* 4.1 *hold, that is, the kth sample-path function $\overline{y}_{m_k}(x;\underline{\omega}_k)$ is increasing and conservative, and there exists X_k^* satisfying $(\overline{y}_{m_k}(x;\underline{\omega}_k) - \gamma)^T(x - X_k^*) > 0$ for all x in some neighborhood of X_k^*. Then, for given positive-valued tolerances ϵ_k and η_k, BA successfully solves the sample-path problem (S_k) for every k w.p.1.*

PROOF. Recall that solving the sample-path problem (S_k) means finding a design polytope P_k so that $v(P_k) \le \epsilon_k$ and image polytope $y P_k$ bounds the target γ to within tolerance η_k.

Following the notation introduced, let $\mathbb{G}_k = \{Z_1, Z_2, \ldots\}$ be the set formed from the points visited by BA during the grafting operations in iteration k. Recall that a grafting operation begins with a segment of length ϵ_k, and always uses segments of increasing size during the search. Also recall that the pruning step is whittling the design line segment obtained after grafting, through a *minimum* number of bisections, to obtain a design line segment whose length is at most ϵ_k. Due to these two facts, there exists $\epsilon > 0$ such that any two adjacent points $Z_j, Z_{j+1} \in \mathbb{G}_k$ satisfy $\|Z_{j+1} - Z_j\| \ge \epsilon$.

This last conclusion implies that if $\mathbb{G}_k$ has an infinite number of elements, either the trajectory of BA: (i) is unbounded, or (ii) has a point where the descent rate is arbitrarily close to or greater than zero. Lemma 4.1 implies that the first possibility occurs with zero probability, while the second cannot happen due to the direction choice rule. Therefore, the set $\mathbb{G}_k$ has a finite number of elements w.p.1. Conclude that BA terminates after a growing step, namely finds a design polytope P_k so that $v(P_k) \leq \epsilon_k$ and its image polytope $y P_k$ bounds the target γ to within η_k w.p.1. □

Theorem 4.4 completes the proof of convergence by showing that if Theorem 1 holds, and if g is assumed to be continuous, then the sequence $\{X_k\}$ of retrospective solutions satisfies $\|g(X_k) - \gamma\| \leq \beta$ for large enough k w.p.1, for arbitrarily small $\beta > 0$. Equivalently, for arbitrarily small $\beta > 0$, $\|X_k - x^*\| \leq \beta$ for large enough k w.p.1.

Theorem 4.4 uses Lemma 4.3, which states that continuous images of shrinking sets also shrink, as long as the sets remain bounded. We only state Lemma 4.3 since its proof is evident from basic results in real analysis.

LEMMA 4.3. *Let $\{\mathfrak{s}_k\}$, $\mathfrak{s}_k \subset \mathbb{R}^q$ be a bounded sequence of sets with diameters converging to zero, namely $\mathfrak{s}_k \subset B(0, r)$ and $\{v(\mathfrak{s}_k)\} \to 0$, where $B(0, r)$ is a ball of radius r centered at the origin. Then, if $g : \mathbb{R}^q \to \mathbb{R}^q$ is a continuous function, $v(g(\mathfrak{s}_k)) \to 0$.*

THEOREM 4.4. *Let the root-finding function g be continuous, strictly increasing, conservative, and have a root x^* satisfying $g(x^*) = \gamma$. Let the sample-path functions $\{\overline{y}_{m_k}(x; \underline{\omega}_k)\}$ be increasing and conservative, and such that there exists X_k^* satisfying $(\overline{y}_{m_k}(x; \underline{\omega}_k) - \gamma)^T (x - X_k^*) > 0$ for all x in some neighborhood of X_k^*. Let the positive-valued sequences $\{\epsilon_k\}$, $\{\eta_k\}$ satisfy $\{\epsilon_k\} \to 0$ and $\{\eta_k\} \to \eta > 0$, with $\eta_k \geq \eta$ for every k.*

If $\{X_k\}$ is the sequence of retrospective solutions, and $\beta > 0$ is a given arbitrarily small number, then

(i) $\|g(X_k) - \gamma\| \leq \beta$ *for large enough k w.p.1;*
(ii) $\|X_k - x^*\| \leq \beta$ *for large enough k w.p.1.*

PROOF. Recall that P_k is the polytope identified by BA at the end of the kth iteration. Since $v(P_k) \leq \epsilon_k$ w.p.1, and $\{\epsilon_k\} \to 0$, we know that $v(P_k) \to 0$ w.p.1.

We next note that the sequence of polytopes $\{P_k\}$, and the corresponding sequence of retrospective solutions $\{X_k\}$, should remain bounded (not necessarily uniformly) w.p.1; for otherwise, we can have an *unbounded* sequence of shrinking polytopes $\{P_k\}$, with each of the image polytopes $y P_k$ bounding the target γ. This is impossible, however, since $\{\overline{y}_{m_k}(x; \underline{\omega}_k)\}$ converges uniformly to g w.p.1, and g has a unique root at x^*.

Since we have just argued that $\{P_k\}$ remains bounded w.p.1, and since $\{v(P_k)\} \to 0$ w.p.1, any realized sequence of $\{P_k\}$ falls within the purview of Lemma 4.3, and hence $\{v\left(g(P_k)\right)\} \to 0$ w.p.1. The sample-path functions $\{\overline{y}_{m_k}(x; \underline{\omega}_k)\}$ converge to the function g, however, and hence $v(y P_k) \to 0$ w.p.1.

The image polytope yP_k bounds the target γ to within η_k for large enough k w.p.1. So, using Lemma 3.3,

$$\|\overline{y}_{m_k}(Z_i;\underline{\omega}_k) - \gamma\| \le (\sqrt{q}+1)(v(yP_k) + \eta_k), \tag{4}$$

where Z_i is chosen from the set $\{Z_0, Z_1, \ldots, Z_n\}$ that forms the design polytope P_k. Since we have argued that $v(yP_k) \to 0$, and since $\eta_k \to \eta$, Inequality (4) implies that

$$\|\overline{y}_{m_k}(Z_i;\underline{\omega}_k) - \gamma\| \le (\sqrt{q}+1)3\eta \text{ for large enough } k \text{ w.p.1.} \tag{5}$$

If X_k is any point in the polytope $P_k \equiv \{Z_0, Z_1, \ldots, Z_n\}$, the triangle inequality implies

$$\|\overline{y}_{m_k}(X_k;\underline{\omega}_k) - \gamma\| \le \|\overline{y}_{m_k}(X_k;\underline{\omega}_k) - \overline{y}_{m_k}(Z_i;\underline{\omega}_k)\| + \|\overline{y}_{m_k}(Z_i;\underline{\omega}_k) - \gamma\|. \tag{6}$$

Conclude from inequalities (5) and (6), and since $v(yP_k) \to 0$ w.p.1, that $\|\overline{y}_{m_k}(X_k;\underline{\omega}_k) - \gamma\| \le (\sqrt{q}+1)4\eta$ for large enough k w.p.1. For given $\beta > 0$, we see that assertion (i) holds upon choosing $\eta = \beta/(4(\sqrt{q}+1))$, and noting that $\{\overline{y}_{m_k}(x;\underline{\omega}_k)\} \to g(x)$ uniformly w.p.1.

The function g is strictly increasing, and hence has a well-defined inverse g^{-1}. Furthermore, since the sequence of retrospective solutions $\{X_k\}$ remains bounded w.p.1, g^{-1} is continuous in this bounded region as well. Conclude from the definition of continuity, and noting that $\beta > 0$ is arbitrary in assertion (i), that assertion (ii) holds. □

Although weaker than the usual notions of convergence to a desired point, the assertions of Theorem 4.4 allow us to obtain a *subsequence* of retrospective solutions that converge to the true root x^* w.p.1.

Having established convergence, it is also interesting to ask if any sort of guarantees can be provided in terms of progress toward the true root x^* of the root-finding function g across iterations. The directions chosen by BA, during the kth iteration, ensure that "some progress" is being made on the kth sample-path function $\overline{y}_{m_k}(x;\underline{\omega}_k)$. Since the sample-path functions $\{\overline{y}_{m_k}(x;\underline{\omega}_k)\}$ converge to the root-finding function g, we should expect that "some progress" be made even with respect to g, at least as k becomes larger. We now characterize this more rigorously.

Define the *descent rate* $h(s)$ at any point $r_k(s)$ in the trajectory as the projection of the vector $\gamma - g(r_k(s))$ in the "positive" direction of the trajectory. Formally,

$$h(s) = (\gamma - g(r_k(s)))^T \frac{r_k^{'+}(s)}{\|r_k^{'+}(s)\|}$$

where

$$r_k^{'+}(s) = \lim_{t \downarrow 0} \frac{r_k(s+t) - r_k(s)}{t}.$$

The descent rate $h(s)$ as defined previously exists for all $s \in [0, l_k)$ because the trajectory $r_k(s)$, $0 \le s \le l_k$ is a piecewise-linear curve.

The trajectory $r_k(s)$, $0 \le s \le l_k$ is a *descent path* if the descent rate $h(s) \le 0$ for all $s \in [0, l_k)$. Equivalently, $r_k(s)$, $0 \le s \le l_k$ is a *descent path* if the line integral

of the vector $g(r_k(s)) - \gamma$ along any portion of the trajectory $r_k(s), 0 \leq s \leq l_k$ is negative, namely, if $\int_{[s_1,s_2]} \left(g(r_k(s)) - \gamma\right)^T \mathbf{dr} \leq 0$ for all $0 \leq s_1, s_2 \leq l_k$. This is essentially because g is continuous and strictly increasing.

Theorem 4.5 proves that in Bounding RA algorithms, except for a finite number of iterations, the trajectory $r_k(s), 0 \leq s \leq l_k$ is a descent path w.p.1.

THEOREM 4.5. *Let the root-finding function g be continuous, strictly increasing, conservative, and have a root x^* satisfying $g(x^*) = \gamma$. Let the sample-path functions $\{\overline{y}_{m_k}(x; \underline{\omega}_k)\}$ be increasing and conservative, and such that there exists X_k^* satisfying $(\overline{y}_{m_k}(x; \underline{\omega}_k) - \gamma)^T (x - X_k^*) > 0$ for all x in some neighborhood of X_k^*. Let the positive-valued sequences $\{\epsilon_k\}$, $\{\eta_k\}$ satisfy $\{\epsilon_k\} \to 0$ and $\{\eta_k\} \to \eta > 0$, with $\eta_k \geq \eta$ for every k. Then, the trajectory $r_k(s), 0 \leq s \leq l_k$ of BA during the kth iteration of the Bounding RA algorithms is a descent path for large enough k w.p.1. Furthermore, there exists $\alpha > 0$ such that for large enough k, the descent rate at each point of the trajectory is at least α.*

PROOF. See online Appendix. □

5. POSTSCRIPT ON KEY ASSUMPTIONS

In this section, we discuss *increasing* and *conservative* function classes. Specifically, we list important properties of these classes, and provide additional intuition on function structure, in order to clarify the subset of SRFPs encompassed by the established convergence results.

Our discussion is deliberately generic since it equally applies to the sample-path function $\overline{y}_{m_k}(x; \underline{\omega}_k)$ and to the underlying function g. We find it remarkable that in most applications involving SRFPs that we have encountered, any function structure in the underlying function g is "shared," almost invariably, by the sample-path function $\overline{y}_{m_k}(x; \underline{\omega}_k)$. So, any discussion on when g may assume a certain structure is implicitly a discussion on the sample-path function as well. There is, however, one notable exception: When $g(x)$ represents a probability expressed as a continuous function of some parameter x, the sample-path function $\overline{y}_{m_k}(x; \underline{\omega}_k)$ tends to be a step function representing the corresponding cumulative frequency.

5.1 Increasing Functions

A function $f : \mathbb{R}^q \to \mathbb{R}^q$ is *increasing* if for $x_1, x_2 \in \mathbb{R}^q$, $(f(x_1) - f(x_2))^T (x_1 - x_2) \geq 0$. It is called *strictly increasing* if the inequality holds strictly whenever $x_1 \neq x_2$. The definitions, expectedly, reduce to the usual notions of increasing and strictly increasing functions when $q = 1$. Many important papers on stochastic root finding, including the original work by Robbins and Monro [1951], and the work by Chen and Schmeiser [2001] on one-dimensional Bounding RA, assume some variation of such a monotonicity assumption on the root-finding function g. The justification for such stringency on the root-finding function is that the class of increasing functions provide a useful and large class for root finding, especially when seen as the derivative of convex functions. Furthermore, the global structure of increasing functions affords the

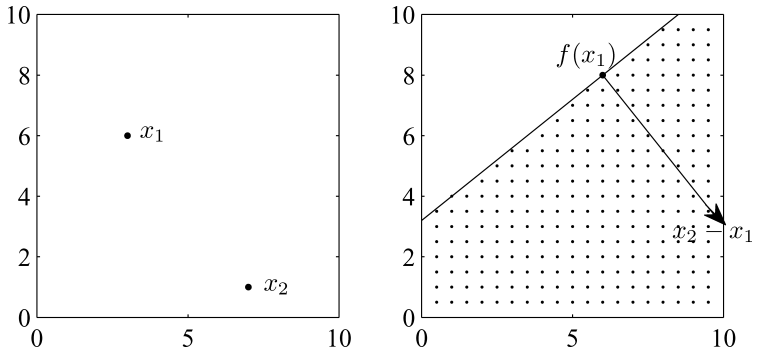

Fig. 6. An increasing function $f : \mathbb{R}^q \to \mathbb{R}^q$ satisfies $(f(x_1) - f(x_2))^T(x_1 - x_2) \geq 0$ for all x_1, x_2. In this figure, the point $f(x_2)$ is stipulated to lie anywhere in the shaded half-space formed by the plane normal to $x_2 - x_1$. Loosely speaking, in increasing functions, the vector $x_2 - x_1$ approximates the direction of $f(x_2) - f(x_1)$ to within an angle $\pi/2$.

ability to develop very efficient algorithms, without assuming the presence of any local structure such as differentiability.

When $q > 1$, the definition $(f(x_1) - f(x_2))^T(x_1 - x_2) \geq 0$ does not provide nearly as much information as in one dimension. The definition only stipulates that in going from x_1 to x_2 the direction of the change in the function values makes an angle less than $\pi/2$ with the vector $x_2 - x_1$. Loosely speaking, this means that having observed the value of $f(\cdot)$ at x_1, the value of $f(\cdot)$ at x_2 can be "guessed" to within a half-space, as depicted in Figure 6. This is often the case in actual applications, where it is usually possible to (only) approximately predict the direction of change of a vector (of performance measures), caused by a change in a vector of designs.

Increasing and strictly increasing functions need not be continuous, and include the subgradients of convex and strictly convex functions, respectively. Increasing functions need not be one to one, and hence include step functions which frequently arise in contexts where the performance measures are probabilities estimated through cumulative fractions. We provide one such example in the next section on numerical examples. Strictly increasing functions are one-to-one functions.

5.2 Conservative Functions

A function $f : \mathbb{R}^q \to \mathbb{R}^q$ is *conservative* if the line integral of f around any closed loop is zero, namely, $\oint f^T \mathbf{dr} = 0$. Equivalently, the function f is conservative if there exists a function h such that $\nabla h = f$. In other words, conservative functions are functions that can be expressed as the gradient of some real-valued function. The function h is then called a harmonic function, or a scalar potential function.

Such a "gradient" interpretation of conservative functions is very useful in our context. It implies that all SRFPs arising as the first-order conditions for optimality in stochastic optimization problems have root-finding and sample-path functions that are conservative by definition. Examples include stochastic optimization problems where the gradient of the objective function can be observed

using one of the many existing techniques for gradient estimation such as infinitesimal perturbation analysis, or the likelihood ratio method [Fu 2006].

Also, a function f is conservative, with a convex scalar potential function h, if it is *differentiable* and *increasing*. This property is a consequence of various results in, Ortega and Rheinboldt [1970, Chapters 5 and 6]. It is relevant to our context because if the root-finding function g is differentiable, we are guaranteed that it is conservative since we have already assumed that it is strictly increasing.

We have assumed that the sample-path functions are conservative, and conservative functions need not be continuous, and the scalar potential function that generates the conservative function can have nondifferentiabilities in a set of measure zero. An example is the subgradient function of a piecewise-linear convex function. It is noteworthy that ASP cannot directly handle functions of this sort in a stochastic root-finding context. The standard assumption in ASP is that the root-finding function be at least twice differentiable.

How does Bounding RA behave when the sample-path functions are not conservative? This is a useful question to answer, since checking whether a function is conservative is often difficult when not expressed in analytic form, or not expressed as the gradient of a real-valued function. Theoretically, BA can be made to cycle indefinitely on sample-path functions that are not conservative. However, in the numerical tests that we have conducted thus far, we have not encountered even a single problem instance where such behavior has been observed.

6. POSTSCRIPT ON CONNECTIONS TO NELDER-MEAD SEARCH

It is worth noting that the proposed Bounding RA algorithm is similar to the famous Nelder-Mead (NM) search [Nelder and Mead 1965] in one important aspect: Both of these algorithms use polytopes as part of their machinery. We believe, however, that the similarities do not extend much further. First, the proposed Bounding RA algorithm is for SRFPs, whereas NM's stochastic variants [Barton and Ivey, Jr. 1996; Humphrey and Wilson 2000] are for SO problems. It is true that all SRFPs can be formulated as SO problems. For example, the root-finding problem of finding x such that $g(x) = 0$, can be recast as finding the unique minimum of the function $f(g(x))$, where $f(y)$ is some chosen function with a unique minimum at $y = 0$. This approach, although correct in principle, is pursued only infrequently. This is because when the chosen function f is differentiable, thereby rendering direct-search techniques less attractive [Swann 1972], the first-order conditions for the transformed optimization problem simply give back another root-finding problem. An interesting line of enquiry is thus identifying situations where a nondifferentiable function f is the natural choice for the transformed problem. In such situations, function minimization through direct-search techniques may become a competitive alternative.

Another key difference between Bounding RA and NM's variants is that the latter maintain polytopes throughout the search process, and continually update them through a rich array of operations, resulting in very

flexible polytopes. By contrast, although Bounding RA also uses polytopes, they are more of a mechanism to ascertain proximity to a solution, and subsequent termination of an RA iteration.

Finally, and probably most importantly, polytope construction within Bounding RA is only a small component sitting within the elaborate RA machinery. This is not the case in NM and its variants, where, arguably, the steps involving polytope manipulation are the key elements of the algorithmic logic.

7. IMPLEMENTATION

In this section, we briefly discuss some implementation strategies in Bounding RA. In Sections 7.1 and 7.2, we focus on providing recommendations for setting the sample-size sequence $\{m_k\}$, and the error-tolerance sequence $\{(\epsilon_k, \eta_k)\}$, in Bounding RA. Other parameters, for example, the step-size multiplier c_2 in BA, are much less important in the sense that Bounding RA's performance seems largely robust to their choice. In Section 7.3, we provide directives on how the RA logic might be terminated in finite time. In Section 7.4, we discuss heuristics to detect conformance to stipulations imposed by convergence.

Our objective is choosing $\{m_k\}$ and $\{(\epsilon_k, \eta_k)\}$ to ensure efficient performance across a broad range of problems, while staying within the confines of conditions imposed by established convergence results. To this extent, some of the recommendations we make are "heuristic." As an example, consider choosing the error-tolerance sequence $\{\epsilon_k\}$. We can show rigorously that it is advantageous to choose $\epsilon_k = O(1/\sqrt{m_k})$, where m_k is the chosen sample size during the kth iteration. The "best" $O(1/\sqrt{m_k})$ sequence, however, is problem dependent and difficult to characterize in advance. Therefore, we limit ourselves to providing a heuristic for choosing a specific $O(1/\sqrt{m_k})$ sequence, based on the physical interpretation of ϵ_k and our numerical experience. The same applies when choosing the error-tolerance sequence in the image space $\{\eta_k\}$.

7.1 Choosing $\{m_k\}$, $\{\epsilon_k\}$

Recall that $\{m_k\}$ is the sequence of sample sizes used to generate the sample-path problems across iterations. The magnitude of m_k thus largely determines how closely the kth sample-path problem approximates the true problem. Likewise, ϵ_k, being the error tolerance in the design space, dictates how much effort is expended in solving the kth sample-path problem. It should thus be expected that the sequences $\{\epsilon_k\}$ and $\{m_k\}$ be chosen "in balance," so that they do not converge to their respective limits, namely zero and infinity, at widely differing rates.

What should be the precise nature of the relationship between $\{m_k\}$ and $\{\epsilon_k\}$ to ensure that the resulting retrospective solutions converge in some optimal fashion? We answer this question in great detail in Pasupathy [2007], where we show that only sequences $\{m_k\}$, $\{\epsilon_k\}$ falling within a certain precisely characterized class $\mathfrak{C}$ ensure that the *generalized mean squared error* (i.e., the expected product of total computational effort and squared error in the retrospective solution) remains finite asymptotically. Intuitively, the "optimal" class

$\mathfrak{C}$ bounds the deviation between the rates at which $\{m_k\}$ and $\{\epsilon_k\}$ converge to their respective limits.

We also show in Pasupathy [2007] that a specific subclass of sequence pairs falling within $\mathfrak{C}$ is

$$m_k = c_1 m_{k-1}, 0 < \limsup_k \epsilon_k m_k < \infty, m_1 = 1, \tag{7}$$

where $c_1 > 1$. If the sample size is constrained to be a positive integer, the expression for m_k can be rounded up or down appropriately. The choice (7) implies that the sample size is increased by a fixed percentage $(100(c-1))$ during each iteration. Bounding RA's performance seems largely robust with respect to the choice of c_1. We demonstrate this through an example in the next section.

In choosing the sequence $\{\epsilon_k\}$, we recall that ϵ_k represents the error tolerance in the design space during the kth iteration. It should thus be chosen as some measure of the uncertainty in the kth root estimate $\overline{X}_k$. One possible choice is

$$\epsilon_k = \sqrt{\mathrm{Tr}\left(\mathrm{Var}(\overline{X}_k)\right)}, \tag{8}$$

where $\mathrm{Tr}\left(\mathrm{Var}(X_k)\right)$ is the trace of the matrix $\mathrm{Var}(\overline{X}_k)$. If we assume that the variance of the retrospective solution X_k takes the canonical form $m_k^{-1}\Sigma$, where Σ is some variance parameter matrix, the retrospective solutions $X_i, i = 1, 2, \ldots$ are independent, and $\overline{X}_k = \sum_{i=1}^k m_i X_i / \sum_{i=1}^k m_i$, then

$$\mathrm{Var}(\overline{X}_k) = \left(\sum_{i=1}^k m_i\right)^{-1} \Sigma = \frac{(c_1 - 1)}{c_1^k - 1}\Sigma. \tag{9}$$

The variance parameter matrix Σ appearing in (9) can be estimated, after $k-1$ iterations, as

$$\hat{\Sigma} = \frac{\sum_{i=1}^{k-1} m_i (X_i - \overline{X}_{k-1})(X_i - \overline{X}_{k-1})^T}{k-2}. \tag{10}$$

The expressions (9) and (10) together provide an estimator for ϵ_k in (8).

7.2 Choosing $\{\eta_k\}$

Just as ϵ_k is a measure of uncertainty in the design space, η_k is a measure of uncertainty in the image space. This prompts us to use

$$\eta_k = \sqrt{\mathrm{Tr}\left(\mathrm{Var}\left(\overline{y}_{m_k}(\overline{X}_{k-1}; \underline{\omega}_k)\right)\right)}. \tag{11}$$

We note that $\overline{X}_{k-1}$ is the root estimate at the end of $k-1$ iterations, and hence forms the initial guess for the kth iteration. The entries of the matrix $\mathrm{Var}(\overline{Y}_{m_k}(\overline{X}_{k-1}))$ can thus be estimated soon after the simulation is executed at the design point $\overline{X}_{k-1}$ in the beginning of the kth iteration.

7.3 "Was-It-An-Accident" Termination Rules

Effective termination rules are problem-, user-, and context-specific. For example, the termination rules suggested here were not used in Section 8, since in

that context our objective is assessing RA's performance as a function of the amount of computing. Nevertheless, a generic termination rule can be useful for some users.

We suggest here "was-it-an-accident" termination rules, which can be used with any stochastic root-finding algorithm when the user needs a solution $\overline{X}_k$ satisfying $g(\overline{X}_k) \in A$, where A is a user-specified region that contains the target γ. Such rules are applied whenever the algorithm produces a candidate solution $\overline{X}_k$ whose response $\overline{y}_{m_{k+1}}(\overline{X}_k; \underline{\omega}_k)$ falls within A; this event is the *it* in "was-it-an-accident." Whenever applied, a side experiment is performed at $\overline{X}_k$ to estimate the probability of the event $\overline{Y}_{m_{k+1}}(\overline{X}_k) \in A$. If the probability is less than a user-specified confidence $1-\alpha$, the event is deemed to have been an accident and the algorithm continues; otherwise, the algorithm terminates.

In our implementation A is a rectangle containing points $\{y = (y_1, y_2, \ldots, y_q) : |y_1 - \gamma_1| \leq \delta_1, |y_2 - \gamma_2| \leq \delta_2, \ldots, |y_q - \gamma_q| \leq \delta_q\}$, where the user-specified vector $\delta = (\delta_1, \delta_2, \ldots, \delta_q)$. Though our multivariate normality assumption next given suggests that the termination rule could be based on an elliptical error probability, we use coordinate-based tolerances δ because they are more easily provided by the user than the parameters of the corresponding ellipsoid centered at γ.

The side experiment could be performed by repeatedly calling the simulation to estimate the probability $\Pr\{\overline{Y}_{m_{k+1}}(\overline{X}_k) \in A\}$. That computing effort, however, could be used instead to further the algorithm. Therefore, the termination rule to follow makes a multivariate-normal assumption on $\overline{Y}_{m_{k+1}}(\overline{X}_k)$, estimates its moments, and then approximates the probability $\Pr\{\overline{Y}_{m_{k+1}}(\overline{X}_k) \in A\}$ using a Monte Carlo side experiment. Such a side experiment, typically performed using the Cholesky decomposition [Law and Kelton 2000], requires negligible computing compared to directly estimating the probability $\Pr\{\overline{Y}_{m_{k+1}}(\overline{X}_k) \in A\}$ by executing the simulation $n \times m_{k+1}$ times at $\overline{X}_k$.

"Was-it-an-accident" logic. At the beginning of the $(k+1)$st iteration, the algorithm has a candidate solution $\overline{X}_k$, a realization $\overline{y}_{m_{k+1}}(\overline{X}_k; \underline{\omega}_k)$ of $\overline{Y}_{m_{k+1}}(\overline{X}_k)$, sample covariance $\hat{\Sigma}$, user-specified acceptance region A, and user-specified confidence $1-\alpha$.

(1) Generate Monte Carlo observations $\overline{y}_1, \overline{y}_2, \ldots, \overline{y}_n$ of $\overline{Y}$, where $n = 100(1-\alpha)/\alpha$ and $\overline{Y} \sim N(\overline{y}_{m_{k+1}}(\overline{X}_k; \underline{\omega}_k), \hat{\Sigma}/m_{k+1})$.

(2) Compute $\widehat{\Pr}\{\overline{Y} \in A\} = n^{-1} \sum_{i=1}^{n} I(\overline{y}_i \in A)$.

(3) Terminate if and only if $\widehat{\Pr}\{\overline{Y} \in A\} \geq 1-\alpha$.

The confidence $1-\alpha$ in "was-it-an-accident" termination rules has both a Bayesian and frequentist interpretation. The likelihood function is assumed to be multivariate normal. If the prior distribution is assumed to be multivariate normal with infinite covariance, then the posterior distribution is the likelihood function. If the algorithm is converging then "was-it-an-accident" rules will eventually terminate, because as the sample size m_k increases the likelihood function's covariance matrix becomes smaller.

7.4 Strategies for Detecting Conformance to Assumptions

Recall that in our convergence proofs (e.g., Theorem 4.5), we have assumed that the individual sample-path problems are increasing, conservative, and possessing a "crossing root" X_k^* satisfying $(\overline{y}_{m_k}(x;\underline{\omega}_k)-\gamma)^T(x-X_k^*)$ for all x in some neighborhood of X_k^*. Furthermore, we have assumed that the underlying root-finding function is strictly increasing with a root x^* satisfying $g(x^*)=\gamma$.

How can we detect that a problem at hand adheres to these stipulations? Unfortunately, there exists no definitive answer to this question. We are thus limited to diagnostic tests, two of which we detail next, to check if the stipulations required for convergence are satisfied.

(D.1) For a given sample-path problem, find a "large" simplex which most likely encloses a "crossing root" X_k^* of the sample-path problem, and check if its image bounds the target γ. Specifically, identify points $X_1, X_2, \ldots, X_{q+1}$ such that, if there exists a solution X_k^* to the sample-path problem, it will reside in the simplex. Then obtain the images $\overline{y}_{m_k}(X_1;\underline{\omega}_k), \overline{y}_{m_k}(X_2;\underline{\omega}_k), \ldots, \overline{y}_{m_k}(X_{q+1};\underline{\omega}_k)$ and check if the polytope $yP = \{\overline{y}_{m_k}(X_1;\underline{\omega}_k), \overline{y}_{m_k}(X_2;\underline{\omega}_k), \ldots, \overline{y}_{m_k}(X_{q+1};\underline{\omega}_k)\}$ bounds the target γ.

(D.2) Sample n points $X_1, X_2, \ldots, X_n$ uniformly in a "region of interest," and obtain their images $\overline{y}_{m_k}(X_1;\underline{\omega}_k), \overline{y}_{m_k}(X_2;\underline{\omega}_k), \ldots, \overline{y}_{m_k}(X_n;\underline{\omega}_k)$. Then for each of the $n(n-1)/2$ pairs $X_i, X_j, i \neq j$ check if the monotonicity assumption holds, namely, if $\left(\overline{y}_{m_k}(X_i;\underline{\omega}_k)-\overline{y}_{m_k}(X_j;\underline{\omega}_k)\right)^T(X_i-X_j)\geq 0$.

Admittedly, the aforesaid diagnostic tests are "heuristic" in that they are based on necessary but not sufficient conditions, and rely on judgement to a certain degree. The situation in practice, however, is less bleak, and the recommended diagnostic tests can be usefully employed.

How does the proposed algorithm behave when the stipulations imposed by convergence are violated? It is our experience that the class of problems characterized by convergence stipulations are much smaller than the class of problems on which the proposed algorithm "works." So, Bounding RA seems to converge even when there are mild violations of the stipulations imposed by the convergence results. This said, one of two symptoms are exhibited when Bounding RA does not converge: (i) In solving the sample-path problems, the iterates progressively step outward, becoming larger and larger in Euclidean norm, thus leading to overflow problems during implementation; or (ii) in solving the sample-path problems, the iterates seem to endlessly cycle in a region without being able to find a bounding polytope. Although both (i) and (ii) are theoretically possible, it has been much easier to devise problems which produce the behavior described in (i) than in (ii).

8. NUMERICAL EXAMPLES

In this section we present two numerical examples to assess the performance of Bounding RA, and to compare it with that of SPSA [Spall 2000], also called Adaptive Simultaneous Perturbation (ASP). An additional example is provided in the online Appendix. For Bounding RA, the error-tolerance sequences in all

examples were chosen according to recommendations in Section 7. Also, in constructing Figures 7–12, enough macroreplications were performed to ensure that the estimated coefficient of variation of the estimated performance measure was less than 0.1.

8.1 Numerical Example I

The example we consider now is a two-dimensional SRFP with root-finding function

$$g(x_1, x_2) = (g_1(x_1, x_2), g_2(x_1, x_2))^T = \left(2x_1^3 + x_1x_2^2, x_2^3 + x_1^2x_2\right)^T,$$

target $\gamma = (1, 1)^T$, simulation response $(Y_1(x_1, x_2), Y_2(x_1, x_2))^T$, where $Y_1(x_1, x_2)$ and $Y_2(x_1, x_2)$ are independent and normally distributed with $\mathrm{E}[Y_1(x_1, x_2)] = g_1(x_1, x_2)$, $\mathrm{E}[Y_2(x_1, x_2)] = g_2(x_1, x_2)$, and each random variable has variance 5. The only solution to the problem is $x^* \sim (0.63908, 0.86481)^T$. The function $g(x_1, x_2)$ is well behaved except for its vanishing first and second derivatives at $(0, 0)^T$.

In assessing algorithm performance, we note that any single run of ASP or Bounding RA is *stochastic* since different runs of the algorithm are essentially different realizations of a stochastic process that corresponds to the solutions generated across iterations. Therefore, any one run may not be truly indicative of overall algorithm performance. To thwart this difficulty, we follow some of the guidelines laid out in Pasupathy and Henderson [2006], and use the distribution of the error as the measure of algorithm performance. Specifically, if X_b is the random variable denoting the returned solution after expending b simulation calls by an algorithm, we are interested in the distribution of one of the random variables $\|X_b - x^*\|$ or $\|g(X_b) - \gamma\|$, where x^* is the true root. For the specific example considered in this section, since the true root x^* is known, the quantiles of the distribution of $\|X_b - x^*\|$ can be estimated without much difficulty.

Recall that ASP has the following iterative structure. Specifically,

$$X_{k+1} = X_k - a_k(\hat{H}(X_k))^{-1}(\overline{Y}_k - \gamma), \tag{12}$$

for $k = 1, 2, \ldots$, where X_0 is an initial guess of the true root x^*, $\overline{Y}_k = \sum_{i=1}^{m} Y_i(X_k)/m$, $\hat{H}(X_k)$ is a matrix serving as the gradient estimate of g at X_k, and $\{a_k\}_{k=0}^{\infty}$ is a predetermined sequence of positive constants. Following the implementation guidelines in Spall [2003, Section 7.8] and Spall [1998, 2000], the required gradient $\hat{H}(\cdot)$ was estimated using symmetric Bernollui direction vectors, and with parameter $\tilde{c}_k$ fixed at $\tilde{c}_k = 1/\sqrt{k+1}$. We used blocking as recommended in Spall [2003, p. 201], with the blocking coefficient $\mathrm{tol}_\theta = 100$.

Figure 7 depicts the performance of ASP for three different step-size sequences $\{a_k\} = \{10/k\}$, $\{a_k\} = \{50/k\}$, and $\{a_k\} = \{100/k\}$, and two different initial solutions $x_0 = x^* = (0.63908, 0.86481)$ and $x_0 = (-100, 100)$. The four curves shown in Figure 7 correspond to 0.85, 0.6, 0.35, and 0.1 quantiles of the distribution of $\|X_b - x^*\|$, where b is the number of simulation calls, and X_b is the solution returned after expending b simulation calls.

We see from Figure 7 that the performance of ASP is sensitive to the choice of parameter settings and the quality of the initial solution. For instance, when the

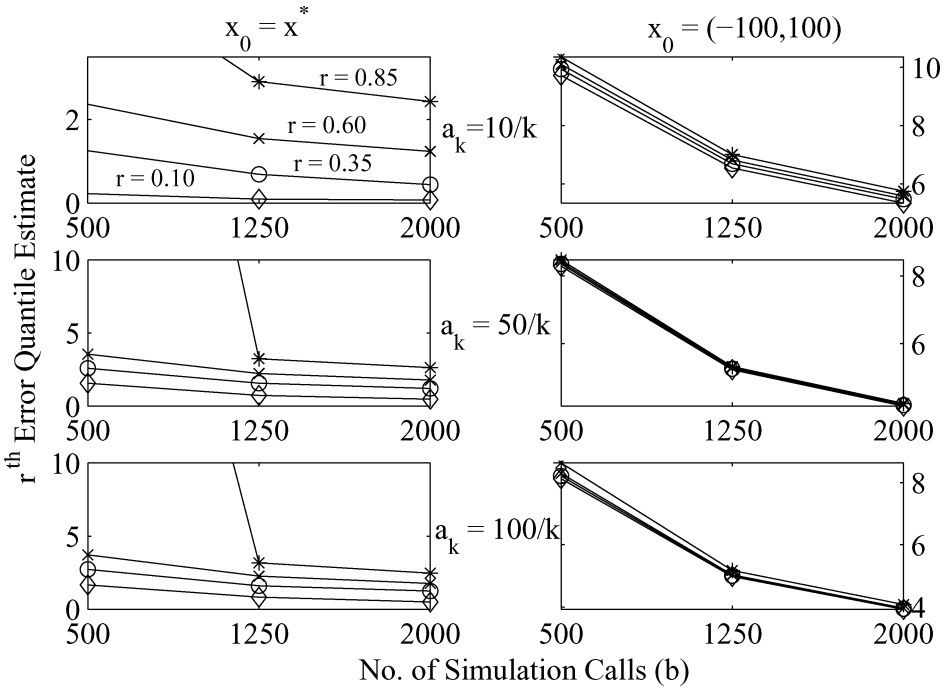

Fig. 7. Performance of ASP for three different parameter settings: (i) $a_k = 10/k$, (ii) $a_k = 50/k$, and (iii) $a_k = 100/k$, and for two different initial solutions: (i) $x_0 = x^* = (0.63908, 0.86481)$, and (ii) $x_0 = (-100, 100)$.

quality of the initial solution is perfect, namely when $x_0 = x^*$, the choice $\{a_k\} = \{10/k\}$ seems to perform much better than the other two choices $\{a_k\} = \{50/k\}$ and $\{a_k\} = \{100/k\}$. When $\{a_k\} = \{50/k\}$ or $\{100/k\}$, even after expending 2000 simulation calls, more than 40% of the solutions generated by ASP are at least 2 units away from the true root x^*. This is in contrast to setting $a_k = 10/k$ when about 75% of the solutions generated by ASP are guaranteed to be less than 2 units from the true root x^* after 2000 simulation calls. This effect is possibly due to the relatively larger step-sizes that result from choosing larger a_k values. The trend is reversed when the initial solution is far away at $x_0 = (-100, 100)$. This is seen in the second column of the graphs in Figure 7, where choosing larger a_k values clearly seems more advantageous, especially after 2000 simulation calls.

From all of our numerical experience with ASP, the blocking parameter tol_θ, which has the effect of artificially "blocking out" iterations in ASP that change the current iterate by "too much," had an important effect on ASP's performance. In Spall [2003, p. 201], it is suggested that this parameter should be "some reasonable maximum distance (tolerance) to cover in one step." A reasonable maximum distance for ASP's iterates, however, is often difficult to determine for a problem, without actually experimenting with various parameter settings. Furthermore, for initial solutions that were much worse, for example, $x_0 = (-10^6, 10^6)$, we were unable to find any combination of parameter settings that produced stable performance consistently.

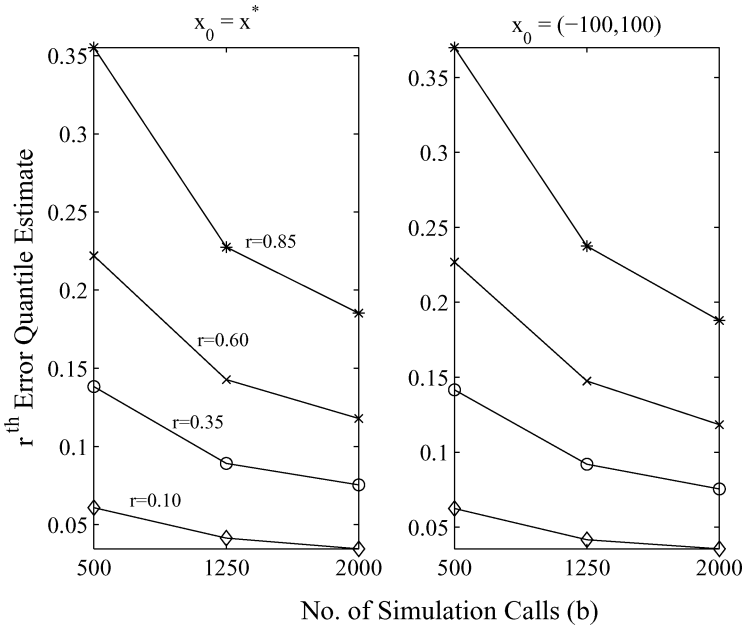

Fig. 8. Performance of Bounding RA with for two different initial solutions: (i) $x_0 = x^* = (0.63908, 0.86481)$ and (ii) $x_0 = (-100, 100)$.

Figure 8 illustrates the performance of Bounding RA on the same problem, with default parameters: initial sample size $m_1 = 1$, initial step size $\delta_0 = 1$, sample-size multiplier $c_1 = 1.1$, and step-size multiplier $c_2 = 2$. As is evident from the figure, the absolute error $\|X_b - x^*\|$ in Bounding RA is an order of magnitude smaller than that in ASP. For example, for both cases $x_0 = x^* = (0.63908, 0.86481)$ and $x_0 = (-100, 100)$, virtually all solutions generated were within 0.25 units, and 35% of the solutions were within 0.07 units, from the true solution x^*, after 2000 simulation calls. Furthermore, the two columns in Figure 8 suggest that the quality of the solutions generated by Bounding RA seem to be largely independent of the initial solution. We elaborate on this last issue in the next numerical example.

8.2 Numerical Example II

In this example, we study Bounding RA's robustness with respect to the quality of the initial solution, dimensionality, and choice of various algorithm parameters.

For the root-finding function $g : \mathbb{R}^q \to \mathbb{R}^q$, we choose the function

$$g(x) = \frac{\sum_{i=1}^{q} 2A_i(x - x^*)e^{(x-x^*)^T A_i(x-x^*)}}{\sum_{i=1}^{q} e^{(x-x^*)^T A_i(x-x^*)}},$$

where A_i is the $q \times q$ matrix with 1 in the $(i, i)^{\text{th}}$ location and zeros everywhere else. We set the target $\gamma = (0, 0, \ldots, 0)$, and the estimator $Y(x) = g(x)+\epsilon$, where ϵ is a q-dimensional Gaussian random variable with mean $(0, 0, \ldots, 0)^T$ and the

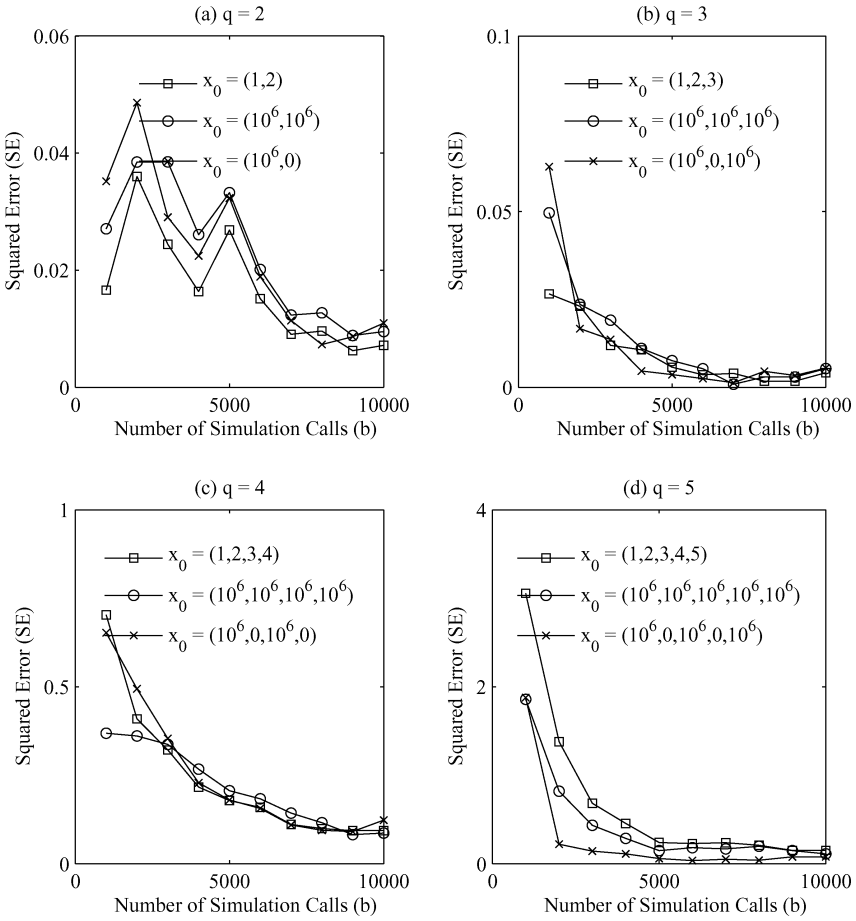

Fig. 9. Squared error of Bounding RA iterates as a function of the number of simulation calls (b) for three initial solutions: (i) $x_0 = (1, 2, \ldots, q)$, (ii) $x_0 = (10^6, 10^6, \ldots, 10^6)$, (iii) $x_0 = (10^6, 0, 10^6, 0, \ldots)$.

$q \times q$ covariance matrix with ones along the diagonal and zeros elsewhere. The true root $x^* = (1, 2, 3. \ldots, q)^T$.

Bounding RA algorithms do not insist that the user provide an initial solution x_0. For convenience in testing, however, we have made x_0 a parameter while fixing the location of the unknown root x^*.

8.2.1 *Dependence on Initial Solution.* Figures 9(a)–(d) show sample plots generated from running the Bounding RA algorithm for three different values of the initial solution x_0, and for dimensions $q = 2, 3, 4, 5$. We plot the squared error $(X_b - x^*)^T(X_b - x^*)$ versus the number of simulation calls b.

Figure 9 results from a single run of the Bounding RA algorithm, that is, a particular choice of random seeds for the simulation calls. To capture the average performance of Bounding RA across sets of random seeds, we plot mean squared error (MSE) of the returned solution X_b as a function of the number of simulation calls b in Figures 10(a)–(d). The curve is generated by averaging

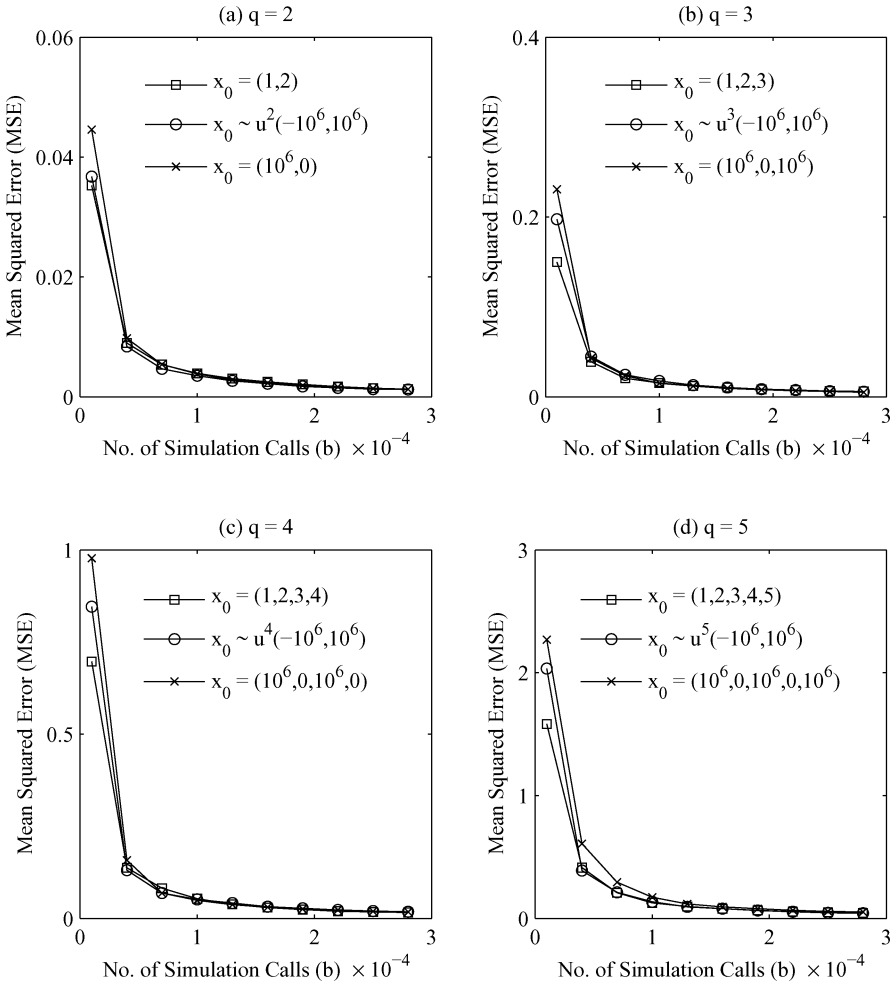

Fig. 10. MSE of Bounding RA iterates versus the number of simulation calls (b) for three initial solutions: (i) $x_0 = (1, 2, \ldots, q)$, (ii) $x_0 \sim u^q(-10^6, 10^6)$, (iii) $x_0 = (10^6, 0, 10^6, 0, \ldots)$.

enough runs of the Bounding RA algorithm to ensure that the sampling error is negligible. In Figure 10, $x_0 \sim u^q(-10^6, 10^6)$ means that the initial solutions were generated uniformly from a q-dimensional hypercube of side 2×10^6 units, centered at the origin.

Figures 9 and 10 suggest two desirable trends. First, irrespective of the location of the initial solution, the generated solutions "converge" to the true root x^* as the expended computing effort increases. Second, the effects of the initial solution are rapidly lost. This behavior is by algorithm design and the result of the iteration-level termination criterion in Bounding RA along with the correct scling of tolerances $\{(\epsilon_k, \eta_k)\}$.

8.2.2 *Dependence on Initial Step-Size, Step-Size Multiplier, and Sample-Size Multiplier.* After choosing parameters in the form suggested in Section 7,

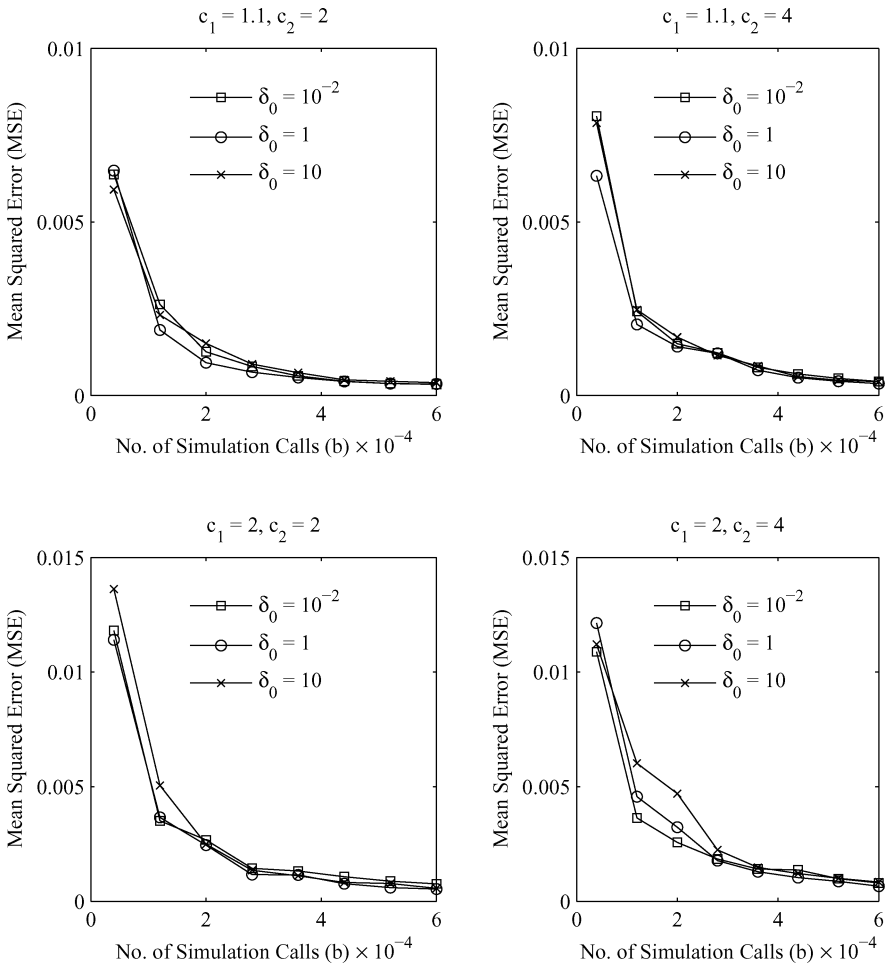

Fig. 11. MSE of Bounding RA iterates versus the number of simulation calls (b) for: (i) initial step-size $\delta_0 = 10^{-2}, 1, 10$; sample-size multiplier $c_1 = 1.1, 2$; and step-size multiplier $c_2 = 2, 4$.

Bounding RA has three remaining parameters: (i) initial step-size δ_0, (ii) sample-size multiplier c_1, and (iii) step-size multiplier c_2. Figure 11 shows the performance of Bounding RA for twelve different combinations of the parameters δ_0, c_1, and c_2. Each panel in Figure 11 corresponds to one of four combinations of c_1 and c_2, and displays three curves corresponding to the three choices $\delta_0 = 10^{-2}$, $\delta_0 = 1$, and $\delta_0 = 10$. We plot the MSE of the returned solution X_b on the vertical axis, and the number of simulation calls b on the horizontal axis.

As is evident from Figure 11, there is no appreciable change in trend as one or more of the parameters δ_0, c_1, and c_2 are altered. If there is a preference for specific parameter values, the choice is based more on convenience rather than performance. For example, we recommend using $c_1 = 1.1$ corresponding to a 10% increase in sample size across iterations, as opposed to using, say, $c_1 = 10$. This is because $c_1 = 1.1$ results in more stopping points and hence

more opportunities for the user to terminate. Likewise we recommend using $\delta_0 = 10^{-2}$ since, loosely speaking, problems are scaled so that solving to 10^{-2} units of precision typically tends to be adequate from a user standpoint.

9. SUMMARY OF RESULTS AND CONCLUDING REMARKS

We now summarize this work and provide some concluding remarks.

(1) A natural generalization of the class of strictly increasing functions in multiple dimensions seems to be $g : \mathbb{R}^q \to \mathbb{R}^q$ satisfying $(g(x_1) - g(x_2))^T(x_1 - x_2) > 0, x_1, x_2 \in \mathbb{R}^q, x_1 \neq x_2$. This class of functions is large (e.g., the gradients of convex functions are strictly increasing as defined) with plenty of underlying structure. In the context of SRFPs, this class is interesting because assuming that the (unknown) root-finding function g is strictly increasing seems to produce a useful class of SRFPs for which specialized algorithms that exploit the underlying structure can be developed.

(2) Bounding RA is a class of RA algorithms that solves SRFPs by identifying a polytope of stipulated diameter whose image under the sample-path function bounds the target γ in a certain precise sense. The diameter stipulation becomes increasingly stringent across iterations. The Bounding RA family of algorithms solves SRFPs when the root-finding function g is continuous, conservative, and strictly increasing in $\mathbb{R}^q$. The sample-path functions are assumed conservative and increasing, and converge uniformly w.p.1 to the root-finding function g.

(3) Our empirical experience suggests that Bounding RA has good finite-time performance that is robust with respect to the choice of algorithm parameters. Particularly, Bounding RA has better finite-time performance than ASP, which is currently the best-known algorithm for solving SRFPs. ASP has good asymptotic properties but algorithm performance is sensitive to initial solution and the choice of algorithm parameters. Both Bounding RA and ASP outperformed CSA in empirical tests.

(4) Even though our theoretical results, and the spirit of our algorithms, are based on the assumption that g is strictly increasing in the sense defined, our numerical experience suggests that the algorithms might work well outside this class. The precise extent to which Bounding RA algorithms can be reliably applied outside this class, however, is yet to be explored.

(5) The Bounding RA algorithm, as presented, is for unconstrained spaces. The most straightforward way to tackle constrained spaces is by reflecting Bounding RA's iterates back into the feasible region whenever they go out of bounds. Another method is to artificially "extend" the sample-path functions from a constrained region to all of $\mathbb{R}^q$. This is easy to do for $q = 1$ but seems to be a difficult problem, in general, for $q > 1$.

ELECTRONIC APPENDIX

The electronic Appendix of this article can be accessed in the ACM Digital Liabrary.

ACKNOWLEDGMENTS

We would like to thank the anonymous referees, the Associate Editor, and the Area Editor. Their comments and suggestions have tremendously helped the content and presentation of various parts of this article.

REFERENCES

ANDRADÓTTIR, S. 1991. A projected stochastic approximation algorithm. In *Proceedings of the Winter Simulation Conference*, B. L. Nelson et al., Eds. Institute of Electrical and Electronics Engineers, Piscataway, NJ, 854–957.

ANDRADÓTTIR, S. 1996. A scaled stochastic approximation algorithm. *Manage. Sci. 42*, 475–498.

ANDRADÓTTIR, S. 1998. A review of simulation optimization techniques. In *Proceedings of the Winter Simulation Conference*, D. Medeiros et. al., Eds. Institute of Electrical and Electronics Engineers, Piscataway, NJ, 151–158.

ATLASON, J., EPELMAN, M. A., AND HENDERSON, S. G. 2004. Call center staffing with simulation and cutting plane methods. *Ann. Oper. Res. 127*, 333–358.

BARTON, R. R. AND IVEY, JR., J. S. 1996. Nelder-Mead simplex modifications for simplex optimization. *Manage. Sci. 42*, 954–973.

BLUM, J. 1954. Multidimensional stochastic approximation. *Ann. Math. Statist. 25*, 737–744.

BROYDEN, C. G. 1965. A class of methods for solving nonlinear simultaneous equations. *Math. Comput. 19*, 577–593.

CHEN, H. 1994. Stochastic root finding in system design. Ph.D. thesis, School of Industrial Engineering, Purdue University, West Lafayette, Indiana.

CHEN, H. AND SCHMEISER, B. W. 2001. Stochastic root finding via retrospective approximation. *IIE Trans. 33*, 259–275.

FABIAN, V. 1968. On asymptotic normality in stochastic approximation. *Ann. Math. Statist. 39*, 1327–1332.

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

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

FU, M. C. 2006. Gradient estimation. In *Simulation*, S. G. Henderson and B. L. Nelson, Eds. Handbooks in Operations Research and Management Science. Elsevier, Amsterdam, Chapter 19, 575–616.

GOLDSMAN, D. AND NELSON, B. 1998. Comparing systems via simulation. In *Handbook of Simulation: Principles, Methodology, Advances, Application, and Practice*. John Wiley & Sons, Chapter 8.

HEALY, K. AND SCHRUBEN, L. W. 1991. Retrospective simulation response optimization. In *Proceedings of the Winter Simulation Conference*, B. L. Nelson et. al., Eds. Institute of Electrical and Electronics Engineers, Piscataway, NJ, 954–957.

HOMEM-DE-MELLO, T. 2003. Variable-Sample methods for stochastic optimization. *ACM Trans. Modeling Comput. Simul. 13*, 108–133.

HUMPHREY, D. AND WILSON, J. R. 2000. A revised simplex search procedure for stochastic simulation response-surface optimization. *INFORMS J. Comput. 12*, 272–283.

JACOBSON, S. H. AND SCHRUBEN, L. W. 1989. A review of techniques for simulation optimization. *Oper. Res. Lett. 8*, 1–9.

KESTEN, H. 1958. Accelerated stochastic approximation. *Ann. Math. Statist. 21*, 41–59.

KUSHNER, H. AND CLARK, D. 1978. *Stochastic Approximation Methods for Constrained and Unconstrained Systems*. Springer, New York.

KUSHNER, H. J. AND YIN, G. G. 2003. *Stochastic Approximation and Recursive Algorithms and Applications*. Springer, New York.

LAW, A. M. AND KELTON, W. D. 2000. *Simulation Modeling and Analysis*. McGraw-Hill, New York.

NELDER, J. A. AND MEAD, R. 1965. A simplex method for function minimization. *Comput. J. 7*, 308–313.

ORTEGA, J. M. AND RHEINBOLDT, W. C. 1970. *Iterative Solution of Nonlinear Equations in Several Variables*. Academic Press, New York.

PASUPATHY, R. 2007. On choosing parameters in retrospective-approximation algorithms for stochastic root finding and simulation optimization. Under review.

PASUPATHY, R. AND HENDERSON, S. 2006. A testbed of simulation-optimization problems. In *Proceedings of the Winter Simulation Conference*, L. Perrone et. al., Eds. Institute of Electrical and Electronics Engineers, Piscataway, NJ.

PASUPATHY, R. K. 2005. Retrospective-Approximation algorithms for the multidimensional stochastic root-finding problem. Ph.D. thesis, School of Industrial Engineering, Purdue University, West Lafayette, Indiana.

PLAMBECK, E. L., FU, B. R., ROBINSON, S. M., AND SURI, R. 1996. Sample-path optimization of convex stochastic performance functions. *Math. Program. 75*, 137–176.

ROBBINS, H. AND MONRO, S. 1951. A stochastic approximation method. *Ann. Math. Statist. 22*, 400–407.

RODRIGUEZ, R. N. 1977. A guide to the Burr type XII distributions. *Biometrika 64,* 1, 129–134.

RUBINSTEIN, R. Y. AND SHAPIRO, A. 1993. *Discrete Event Systems: Sensitivity Analysis and Stochastic Optimization by the Score Function Method*. John Wiley & Sons, New York.

RUSZCZYNSKI, A. AND SHAPIRO, A., EDS. 2003. *Stochastic Programming. Handbook in Operations Research and Management Science*. Elsevier, New York.

SAFIZADEH, M. H. 1990. Optimization in simulation: Current issues and the future outlook. *Naval Res. Logist. 37*, 807–825.

SHAPIRO, A. 1991. Asymptotic analysis of stochastic programs. *Ann. Oper. Res. 30*, 169–186.

SHAPIRO, A. 1996. Simulation based optimization. In *Proceedings of the Winter Simulation Conference*, J. M. Charnes et. al., Eds. Institute of Electrical and Electronics Engineers, Piscataway, NJ, 332–336.

SHAPIRO, A. 2000. Stochastic programming by Monte Carlo simulation methods. *Stochastic Programming E-Print Series*. http://hera.rz.hu-berlin.de/speps/.

SHAPIRO, A. AND HOMEM-DE-MELLO, T. 2000. On the rate of convergence of optimal solutions of Monte Carlo approximations of stochastic programs. *SIAM J. Optimiz. 11,* 1, 70–86.

SPALL, J. C. 1998. Implementation of the simultaneous perturbation algorithm for stochastic optimization. *IEEE Trans. Aerospace Electron. Syst. 34*, 817–823.

SPALL, J. C. 2000. Adaptive stochastic approximation by the simultaneous perturbation method. *IEEE Trans. Autom. Control 45*, 1839–1853.

SPALL, J. C. 2003. *Introduction to Stochastic Search and Optimization*. John Wiley & Sons, Hoboken, NJ.

SWANN, W. H. 1972. *Direct Search Methods*. Numerical Methods for Unconstrained Optimization. Academic Press, London, Chapter 2, 13–28.

VENTER, H. J. 1967. An extension of the Robbins-Monro procedure. *Ann. Math. Statist. 38*, 181–190.

WASAN, M. T. 1969. *Stochastic Approximation*. Cambridge University Press, Cambridge, UK.

YAKOWITZ, S., L'ECUYER, P., AND VAZQUEZ-ABAD, F. 2000. Global stochastic optimization with low-dispersion point sets. *Oper. Res. 48*, 939–950.

Received October 2006; revised June 2008; accepted June 2008