NOTE ON THE EFFECTIVENESS OF STOCHASTIC OPTIMIZATION ALGORITHMS FOR ROBUST DESIGN

Manjula A. Iyer[1], Rhonda D. Phillips[1]
Michael W. Trosset[2], Layne T. Watson[3]

[1]Department of Computer Science
Virginia Polytechnic Institute and State University
Blacksburg, VA, 24061, USA
e-mail: {manjula,rdphllps}@vt.edu

[2]Department of Statistics
Indiana University
Bloomington, IN 47405, USA
e-mail: trosset@indiana.edu

[3]Departments of Computer Science and Mathematics
Virginia Polytechnic Institute and State University
Blacksburg, VA, 24061, USA
e-mail: ltw@cs.vt.edu

Abstract: Robust design optimization (RDO) uses statistical decision theory and optimization techniques to optimize a design over a range of uncertainty (introduced by the manufacturing process and unintended uses). Since engineering objective functions tend to be costly to evaluate and prohibitively expensive to integrate (required within RDO), surrogates are introduced to allow the use of traditional optimization methods to find solutions. This paper explores the suitability of radically different (deterministic and stochastic) optimization methods to solve prototypical robust design problems. The algorithms include a genetic algorithm using a penalty function formulation, the simultaneous perturbation

stochastic approximation (SPSA) method, and two gradient-based constrained nonlinear optimizers (method of feasible directions and sequential quadratic programming). The results show that the fully deterministic standard optimization algorithms are consistently more accurate, consistently more likely to terminate at feasible points, and consistently considerably less expensive than the fully nondeterministic algorithms.
AMS Subject Classification: 65C20, 65K05, 68U99
Key Words: design under uncertainty, genetic algorithm, multidisciplinary design optimization, stochastic optimization

1. Introduction

Uncertainty is prevalent and unavoidable in engineering design, yet modern design methods ignore uncertainty, improperly address uncertainty, or require significant computational resources to effectively consider uncertainty. Uncertainty can be introduced by the manufacturing process that yields a finished product different from the original design and operational use of the actual product that differs from intended use or exceeds the limits of constraints on which the design is based. Statistical decision theory, and the Bayes' principle in particular, provide a means to quantify uncertainty and rigorously incorporate uncertainties in the design process, but this is often computationally expensive.

RDO, an exceedingly challenging class of multidisciplinary design optimization (MDO), considers both noise and design variables [1]. Given the large expense of calculating accurate MDO results, it is computationally infeasible to evaluate functions at more than 100 or so design points [2]. The computational complexity involved in evaluating functions has led to the use of surrogates, which are low cost approximations of both the objective and constraint functions. Methods used include classical response surface approximations (RS) [3], Bayesian estimators known as DACE [4], and variable-complexity modeling (VCM) [5].

Effectively using the Bayes principle to consider uncertainty in design is further hindered by the high computational expense of calculating expectations, which require numerical integrations. Just as surrogates reduce computational complexity associated with function evaluations, it may be possible to use surrogates to effectively reduce the computational complexity associated with

calculating expectations [6]. Reducing the computational cost of robust design techniques will make robust design increasingly accessible, allowing for more reliable, quicker, and less expensive engineering designs.

This paper explores the suitability of radically different optimization techniques to solve prototypical robust design problems. While optimization problem formulations can be either deterministic or stochastic, algorithms used to solve either of the formulations may be deterministic or stochastic as well. There is no intrinsic reason why a deterministic formulation would require use of a deterministic algorithm or a stochastic formulation would require a stochastic algorithm. In fact, certain deterministic optimization problems are often best solved by stochastic algorithms, and vice versa. Furthermore, the same applies to global optimization—there is no inherent reason why nondeterministic algorithms should be more efficient for global optimization than deterministic algorithms.

This work compares the results of two deterministic optimization algorithms and two stochastic optimization algorithms used to solve example robust design problems. Each algorithm is compared in terms of the quality of the resulting designs (feasibility and objective function values) and the amount of work required to achieve an acceptable design (measured in system analyses). Performance could be examined for either progress-based optimization (i.e., algorithm termination based on perceived convergence) or budget-based optimization (i.e., algorithm termination based on a maximum number of allowed system analyses). Results for only the former are presented here—a budget-based analysis will be the topic of future work. In addition to the comparison of optimization algorithms, empirical evidence is gathered to aid engineers in choosing appropriate parameters for the various optimization algorithms.

The remaining sections of this paper are organized as follows. Section 2 provides a brief introduction to robust design optimization (RDO). Section 3 describes the optimization algorithms used, and Section 4 introduces the example problems formulated to test the optimization algorithms' suitability to solve RDO problems. Section 5 contains experimental results and discussion, and Section 6 concludes the paper.

2. Robust Design Optimization

Welch and Sacks, in relation to the design and analysis of computer experiments (DACE), described the class of problems discussed in this paper as quantifying designable engineering parameters and manufacturing process parameters in order to use models to generate quality characteristics [7]. The designable engineering parameters should then be chosen such that the quality of the end result product is good across the variability of manufacturing conditions. This problem predates computer models as Taguchi first proposed an engineering design process in three stages that would evaluate and fine tune a design after noise was considered [8].

Using statistical decision theory techniques, an optimization problem can be formulated for robust design problems. An objective function for a RDO problem has the form $f : A \times B \rightarrow \Re$, where $a \in A$ are the design variables, $b \in B$ are the manufacturing process variables, and $f(a; b)$ represents the loss from operating design a under conditions b. The goal is to find $a^* \in A$ such that, for every $b \in B$,

$$f(a^*; b) \leq f(a; b) \quad \forall a \in A,$$

although this goal may be impossible to realize. Finding $a^* \in A$ that minimizes $f(a; b)$ for each $b \in B$ is addressed in statistical decision theory by finding a decision rule that minimizes risk. Using Bayes principle to minimize average loss, the optimization problem becomes

$$\min_{a \in A} \phi(a),$$

with objective function

$$\phi(a) = \int_B f(a; b) p(b)\, db,$$

where p is a probability density on B and can be customized to fit a particular application such as the probability distribution of random manufacturing errors.

Welch and Sacks proposed applying a numerical optimizer to an objective function of the above form, but with some modifications [7]. Evaluating the function f above is often expensive,

but performing the integration is much more expensive, rendering traditional optimization methods intractable for these problems. Welch and Sacks proposed the following scheme (an instance of well known response surface techniques), referred to as DACE, for optimization of an expensive objective function $\psi : A \to \Re$.

1. Choose parameter points $a_1, \ldots, a_N \in A$.
2. Compute $\psi(a_1), \ldots, \psi(a_N)$.
3. Construct a Bayesian estimator (surrogate) $\hat{\psi}(a)$ for $\psi(a)$.
4. Minimize $\hat{\psi}$ over A.

DACE has gained popularity as an alternative to traditional optimization of computationally expensive objective functions that do not include uncertainty. This work is focused on algorithms for optimization problems involving uncertainty, and builds on a previous work that explored the cost of integration in computing the loss $\phi(a)$ [6]. This work explores the application of various deterministic and stochastic optimization methods to RDO problems, with the ultimate application being to a surrogate objective function in a RS, VCM, or DACE framework.

3. Algorithms

This section contains short descriptions of the optimization methods used to obtain the results listed in Section 5.

Genetic Algorithms. Genetic algorithms (GAs) were first introduced by Holland, and have been used in many fields including medicine, engineering, and business to optimize functions that are not well suited to traditional optimization methods [9][10][11]. There are important differences between GAs and traditional optimization methods. GAs consider entire populations of designs rather than single designs, and each design or individual is represented as an encoded string. GAs do not require gradients and need only objective function values (fitnesses of individuals). GAs are not deterministic as they involve random steps in their search, and they search globally for solutions, which makes this class of algorithms arguably more effective than deterministic exhaustive search for many problems.

The biological rationale is that GAs move toward an optimal solution to an optimization problem in the same manner that evolutionary processes yield superior individuals through natural

selection. Two parent individuals reproduce to form a new individual that preserves and forms new genetic information. Individuals survive based on fitness, which is measured by an objective function, and more fit individuals are more likely to pass on genetic information to the next generation, resulting in a more fit population.

In populations, evolution is simulated by propagation of the species through reproduction and random variations in genetics. Fitness is evaluated based on each individual's objective function value, and rules for survival and reproduction are predetermined to select fitter individuals most often. This search attempts to locate the most fit individual, or the individual that minimizes (or analogously, maximizes) the objective function. Formally, the goal is to approximate a global minimum point $a^* \in A$ of a real-valued objective function $f : A \to \Re$, i.e., $f(a) \geq f(a^*)\ \forall a \in A$.

SPSA. SPSA (simultaneous perturbation stochastic approximation) is an algorithm for stochastic optimization of multivariate systems that relies only on the measurement of the objective function to be optimized [12]. Like GAs, this algorithm does not require the gradient of the objective function. SPSA has been shown to be very effective in high dimensional problems as it provides a reasonable solution using a relatively small number of measurements of the function values.

The most important feature of SPSA is the underlying gradient approximation that requires only two objective function measurements per iteration regardless of the dimension of the optimization problem. These two measurements are made by simultaneously and randomly varying all of the variables in the problem (the "simultaneous perturbation").

Consider the problem of minimizing a differentiable loss function $L(\theta)$, where θ is a p-dimensional vector. The optimization problem is finding the minimizing point θ^* such that the gradient $\nabla L(\theta^*) = 0$. The general form of the SPSA algorithm is

$$\hat{\theta}_{k+1} = \hat{\theta}_k - a_k \hat{g}_k(\hat{\theta}_k),$$

where $\hat{g}_k(\hat{\theta}_k)$ is the estimate of the gradient $g(\theta)$ at the iterate $\hat{\theta}_k$, and the sequence $a_k \downarrow 0$ [12].

Let $y(\cdot)$ represent a measurement of $L(\cdot)$ that contains an error (i.e., $y(\cdot) = L(\cdot)+$ noise) and let c_k be a small positive number.

One-sided gradient approximations require measurements $y(\hat{\theta}_k)$ and $y(\hat{\theta}_k+$ perturbation) and two-sided gradient approximations require measurements $y(\hat{\theta}_k)$ and $y(\hat{\theta}_k\pm$ perturbation).

The simultaneous perturbation approximation has all elements of $\hat{\theta}_k$ randomly perturbed to obtain two measurements of $y(\cdot)$. Each component of the gradient approximation is a ratio involving the individual components in the perturbation vector and the difference in the two corresponding measurements. Each component $\hat{g}_{ki}(\hat{\theta}_k)$, $i = 1, \ldots, p$, for the two-sided simultaneous perturbation is given by

$$\hat{g}_{ki}(\hat{\theta}_k) = \frac{y(\hat{\theta}_k + c_k\Delta_k) - y(\hat{\theta}_k - c_k\Delta_k)}{2c_k\Delta_{ki}},$$

where Δ_k is a p-dimensional random perturbation vector, the distribution of which is specified by the user.

Standard Gradient-based Algorithms. Two gradient-based optimizers, a sequential quadratic programming (SQP) method and a modified method of feasible directions (MMFD), from the Design Optimization Tools (DOT) software (Version 4.0, distributed by Vanderplaats Research & Development) are used to obtain results. Descriptions of SQP and MMFD are available in [13], and [14] includes a comparison of DOT to optimizers commonly used to solve MDO problems.

4. Example Problems

In order to evaluate the given algorithms, two representative robust design problems have been selected. Descriptions of the two example problems follow.

Problem 1. Let $d = (d_1, d_2)^t$ be the design variables, with design bounds $d_1 \in [-10, 10]$ and $d_2 \in [0, 10]$. Let $X = (X_1, X_2)^t$ be the uncertainty, a random vector of unknown physical parameters or operating conditions, and assuming $X_1 \sim \text{Uniform}(-1, 1)$ and $X_2 \sim \text{Uniform}(-3/4, 3/4)$. The following algorithm denotes a "coupled analysis" $y(d, X)$:

$y_2 \leftarrow 5$
Do until convergence:
  $\{y_1 \leftarrow d_1^2 + d_2 - y_2/5 + X_1;$
  $y_2 \leftarrow d_1 - d_2^2 + (1/2)e^{-y_1^2} + X_2\}$
Return (y_1, y_2)

Let

$$f(d, X) = \begin{cases} 1, & \text{if product } d \text{ fails under } X, \\ 0, & \text{otherwise,} \end{cases}$$

where failure is indicated by $y_1(d, X) < 8$ or $y_2(d, X) > 5$. The probability that failure occurs is $E[f(d, X)]$, and $\tau > 0$ indicates a tolerable probability of failure.

Setting aside the prospect of failure, let

$$L(d, X) = (d_1 + 1)^2 + 10d_2^2 + y_1(d, X)$$

represent the loss that occurs as a result of a successful operating design d under condition X. The comparable risk of operating design d is denoted by

$$R(d) = E[L(d, X)] = (d_1 + 1)^2 + 10d_2^2 + E[y_1(d, X)].$$

The optimization problem is

$$\begin{aligned} \min_d R(d) \quad \text{subject to} \quad & E[f(d, X)] \leq \tau, \\ & d_1 \in [-10, 10], \\ & d_2 \in [0, 10]. \end{aligned}$$

Refer to [6] for contour plots of this objective function $R(d)$ and the expected value $E[f(d, X)]$. The design domain contains sizeable regions in which the constraint is flat, where the probability of success or failure of the design is equal to one. In these cases, gradient based optimizers will likely fail if an infeasible design is given in these flat sections because the gradient will evaluate to zero. Furthermore, within these flat regions Monte Carlo based integrators will produce an exact result after one function evaluation. Problem 1 with $\tau = 0.015$ has two local minima: 25.861 at $d \approx (3.104, 0)^t$, and 12.642 at $d \approx (-2.904, 0)^t$.

Problem 2. Problem 2 is a modification of Problem 1 that contains no flat constraint regions in the domain. Changing the distribution of the noise variables from a uniform to a normal removed the flat regions, but unfortunately resulted in a more challenging integrand.

Let $d = (d_1, d_2)^t$ be the design variables, with design bounds $d_1 \in [-10, 10]$ and $d_2 \in [0, 10]$. Let $X = (X_1, X_2)^t$ be the uncertainty, a random vector of unknown physical parameters or operating conditions, and assuming $X_1 \sim N(0,1)$ and $X_2 \sim N(0,1)$. The following algorithm denotes a "coupled analysis" $y(d, X)$:

$y_2 \leftarrow 5$
Do until convergence:
$\{y_1 \leftarrow d_1^2 + d_2 - y_2/5 + X_1;$
$y_2 \leftarrow d_1 - d_2^2 + (1/2)e^{-y_1^2} + 40X_2\}$
Return (y_1, y_2)

Let

$$f(d, X) = \begin{cases} 1, & \text{if product } d \text{ fails under } X, \\ 0, & \text{otherwise,} \end{cases}$$

where failure means $y_1(d, X) > 30$ and $y_2(d, X) < -80$. The probability that failure occurs is $E[f(d, X)]$, and $\tau > 0$ indicates a tolerable probability of failure.

Setting aside the prospect of failure, let

$$L(d, X) = d_1^2 + 10(d_2 - 7)^2 + y_1(d, X)$$

represent the loss that occurs as a result of a successful operating design d under condition X. The comparable risk of operating design d is denoted by

$$R(d) = E[L(d, X)] = d_1^2 + 10(d_2 - 7)^2 + E[y_1(d, X)].$$

The optimization problem is

$$\min_d R(d) \quad \text{subject to} \quad \begin{aligned} & E[f(d, X)] \leq \tau, \\ & d_1 \in [-10, 10], \\ & d_2 \in [0, 10]. \end{aligned}$$

Refer to [6] for contour plots of this objective function $R(d)$ and the expected value $E[f(d, X)]$. Problem 2 with $\tau = 0.015$ has two local minima: 25.240 at $d \approx (0.055, 5.881)^t$, and 416.669 at $d \approx (8.024, 1.636)^t$.

5. Results and Discussion

Results for Problem 1. All results are for the failure tolerance $\tau = 0.015$. For the gradient-based algorithms MMFD and SQP in DOT, the same error tolerances and algorithmic choices as in [6] are used here, for comparison purposes. Results are given for two starting points d, $(-10, 10)^t$ and $(10, 6)$, since the starting point matters for gradient-based algorithms. For Problem 1 with both starting points, DOT SQP failed in all but a couple cases, so Tables 1 and 2 do not give results for SQP.

For the GA, the real variables d_1, d_2 were not encoded, but used directly in real number crossover as described in [15]. Some experimentation was done with the mutation rate and population size, with the best results being obtained for mutation probability 0.01 with population size 20. The crossover probability was 1.0. The GA termination criterion was ten consecutive generations without improvement in the fitness. As described in [15], an elitist selection strategy was used, with parents being chosen by a roulette wheel algorithm based on population rank.

Spall's algorithm SPSA was implemented with $c_k = 0.01$ and $\Delta_k \sim$ Symmetric Bernoulli$(-1, +1)$ (i.e., each component of Δ_k is a random variable $(X - n/2)/(n/2)$, where X is a Bernoulli random variable for $n = 20$ trials with $p = 0.5$), and the termination criterion was $\|\hat{\theta}_{k+1} - \hat{\theta}_k\| \leq 0.001$.

In the tables, DOT MMFD with a quasi-Monte Carlo integration algorithm to compute the expectations is the base case, since it always correctly found a local minimum point. This is a completely deterministic algorithm, since quasi-Monte Carlo integration is effectively deterministic [6]. For comparison, DOT MMFD was also implemented with a true Monte Carlo integrator. GA and SPSA with Monte Carlo integration are both fully nondeterministic, both in the optimization algorithm and in the expectation computation (integrals). In all the tables, "number of points" refers to the number of times the integrand in an expectation integral is evaluated. More points generally means more accurate integrals, and thus more accurate objective and constraint function values. "Reported optimum" is the purported optimum value of $R(d)$ found by the algorithm, with the mean and standard deviation σ from 10 runs. When different runs terminate near different local optimum points, two averages (one for each

local optimum cluster) are reported on separate lines in the tables; the data in the remainder of such a pair of rows applies to all 10 runs. The number of times $y(d, X)$ is evaluated defines the number of "system analyses," which in an MDO problem is the number of converged coupled subsystem analyses that must be completed (here there are two subsystems with state variables y_1 and y_2). "Optimizer calls" refers to the number of times the algorithm calls the subroutine that evaluates $R(d)$ and the constraints; this is more meaningful for DOT than the other algorithms. The last column, "percent feasible," is the percent of the 10 runs that resulted in the final point being actually feasible, where feasibility is determined with high accuracy (within 10^{-8}, using, e.g., an adaptive Newton-Cotes algorithm for the integrals). Since a GA and SPSA cannot deal explicitly with constraints, a penalty function formulation is used:

$$\lim_{\lambda\to\infty} \min_{d\in D} R(d) + \lambda\big(E[f(d, X)] - \tau\big)_+,$$

where $w_+ \equiv \max\{0, w\}$, $D = [-10, 10] \times [0, 10]$, and λ (initially 10^9) is adjusted to ensure feasibility within the accuracy of the other computations.

From Tables 1 and 2 the following observations can be made. As expected, DOT is sensitive to the starting point, and from both starting points DOT converged to the larger local minimum 25.86. The GA and SPSA did get close to the global minimum 12.64 occasionally, but neither the GA nor SPSA were as consistently accurate and feasible as DOT MMFD with the quasi-Monte Carlo integrator. Furthermore, for each number of points (integrand samples per integral calculation), DOT MMFD QMC required far fewer system analyses than both the GA and SPSA. Succinctly, *the GA and SPSA consistently required more work for lower quality results than DOT MMFD QMC.*

Results for Problem 2. All of the general comments earlier about algorithm parameters and table format likewise apply to Tables 3 and 4 for Problem 2 (for which $\tau = 0.015$ also). DOT MMFD frequently failed for Problem 2 [6], so only results for DOT SQP are shown. The dash entries in the tables mean DOT SQP failed to converge. Problem 2 has two local minima, 25.24 and 416.67, sometimes found by different runs of the GA and SPSA.

Table 1. Performance-based results for Problem 1. The initial point $d = \theta_0 = (-10, 10)^t$.

No. Pts.	Reported Optimum Mean(σ)	System Analyses Mean(σ)	Optimizer Calls Mean(σ)	% Feas.
DOT (MMFD), Quasi-MC Integrator				
10^1	25.39	8.80E+02	43.0	0
10^2	25.87	8.80E+03	43.0	100
10^3	25.86	7.60E+04	37.0	100
10^4	25.86	7.80E+05	38.0	100
10^5	25.86	7.40E+06	36.0	100
DOT (MMFD), MC Integrator				
10^1	25.05(0.39)	1.01E+03(2.81E+02)	49.5(14.0)	0
10^2	25.76(0.11)	1.05E+04(1.98E+03)	51.4(9.9)	20
10^3	25.83(0.04)	9.70E+04(1.90E+04)	47.5(9.5)	30
10^4	25.86(0.01)	9.12E+05(2.18E+05)	44.6(10.9)	80
10^5	25.86(0.00)	7.84E+06(8.00E+04)	38.2(0.4)	40
GA, MC Integrator				
10^1	11.99(1.23) 26.12(2.21)	1.53E+04(3.19E+03)	305.4(97.9)	40
10^2	12.12(1.05) 25.78(1.46)	1.69E+05(5.57E+04)	356.7(83.9)	40
10^3	12.26(1.80) 25.89(2.51)	1.87E+06(3.16E+04)	402.8(101.9)	50
10^4	13.20(3.22) 27.16(2.93)	2.40E+07(7.01E+06)	399.1(91.8)	100
10^5	13.66(2.78) 29.73(4.76)	2.55E+08(6.11E+07)	377.6(88.1)	80
SPSA, MC Integrator				
10^1	13.97(4.53)	2.67E+03(8.62E+02)	52.6(10.1)	60
10^2	11.43(5.32)	1.58E+04(6.54E+03)	49.4(7.8)	50
10^3	13.87(1.88) 25.14(3.12)	3.12E+05(7.54E+04)	64.3(9.5)	80
10^4	12.81(2.36) 25.93(1.76)	2.22E+06(5.96E+05)	55.6(7.8)	100
10^5	14.16(2.11) 27.16(2.32)	1.97E+07(3.21E+06)	59.8(9.9)	100

Table 2. Performance-based results for Problem 1. The initial point $d = \theta_0 = (10, 6)^t$.

No. Pts.	Reported Optimum Mean(σ)	System Analyses Mean(σ)	Optimizer Calls Mean(σ)	% Feas.
DOT (MMFD), Quasi-MC Integrator				
10^1	25.39	1.16E+03	57.00	0
10^2	25.87	1.08E+04	53.00	100
10^3	25.86	1.10E+05	54.00	100
10^4	25.86	1.18E+06	58.00	100
10^5	25.86	1.14E+07	56.00	100
DOT (MMFD), MC Integrator				
10^1	22.63(5.26)	1.10E+03(1.89E+02)	54.10(9.45)	10
10^2	25.79(0.04)	1.18E+04(1.51E+03)	58.00(7.56)	40
10^3	25.84(0.04)	1.16E+05(1.89E+04)	57.20(9.43)	40
10^4	25.85(0.01)	1.10E+06(1.35E+05)	54.00(6.74)	20
10^5	25.86(0.00)	1.11E+07(8.35E+05)	54.30(4.17)	20
SPSA, MC Integrator				
10^1	25.97(5.64)	3.86E+03(7.14E+02)	61.2(7.6)	50
10^2	22.43(3.32)	2.45E+04(5.36E+03)	58.6(9.8)	50
	11.96(2.42)			
10^3	14.93(3.66)	4.43E+05(3.68E+04)	51.4(6.5)	70
10^4	12.76(3.46)	3.12E+06(8.23E+05)	55.6(9.6)	50
10^5	12.59(2.11)	2.93E+07(4.41E+06)	61.4(8.9)	50
	25.73(0.00)			

When this occurs, two "mean reported optimum" values are given in the tables in consecutive rows; the data in the remainder of these two rows applies to all 10 runs.

Referring to Table 3, DOT SQP with the quasi-Monte Carlo integrator correctly found the global minimum 25.24 with at least 10^4 points, and always terminated with a feasible point (except for the failures at 10 and 100 points). DOT SQP with the Monte Carlo integrator mostly failed, indicating that SQP is not a robust technique in the presence of noise. For 10^3 points, the GA came close to both local optima, but for more points the GA usually failed to come close to either solution, showing the typical erratic behavior of GAs (admittedly a larger number of runs would have

Table 3. Performance-based results for Problem 2. The initial point $d = \theta_0 = (-10, 10)^t$.

No. Pts.	Reported Optimum Mean(σ)	System Analyses Mean(σ)	Optimizer Calls Mean(σ)	% Feas.
DOT (SQP), Quasi-MC Integrator				
10^1	—	—	—	—
10^2	—	—	—	—
10^3	25.63	1.22E+05	60.0	100
10^4	25.24	9.20E+05	45.0	100
10^5	25.25	9.20E+06	45.0	100
DOT (SQP), MC Integrator				
10^1	255(217)	1.22E+03(6.00E+02)	60.0(30.0)	10
10^2	39.42(0.00)	3.60E+04(0.00E+00)	179.0(0.0)	10
10^3	29.38(3.89)	2.56E+05(1.68E+05)	127.0(84.2)	20
10^4	29.68(0.00)	1.02E+06(0.00E+00)	50.0(0.0)	10
10^5	294(186)	1.35E+07(3.92E+06)	66.7(19.6)	20
GA, MC Integrator				
10^1	19.12(3.59)	1.21E+04(2.22E+03)	420.2(87.9)	40
10^2	25.65(9.97)	3.56E+05(1.58E+04)	499.4(91.5)	50
10^3	24.63(3.21) 417(1.24)	4.78E+06(3.68E+04)	503.4(77.8)	40
10^4	17.83(0.00) 441(17)	2.63E+07(8.85E+05)	456.4(90.1)	80
10^5	23.54(5.74)	5.85E+08(7.76E+07)	401.9(100.2)	80
SPSA, MC Integrator				
10^1	22.23(5.65)	1.43E+03(6.89E+02)	73.7(8.2)	80
10^2	19.48(8.34)	9.86E+03(8.82E+02)	79.5(6.9)	40
10^3	28.97(4.46)	5.44E+04(1.26E+03)	66.3(4.8)	100
10^4	29.43(0.00) 441(10)	4.38E+05(2.22E+04)	68.4(7.6)	80
10^5	25.12(3.86)	2.46E+06(9.55E+04)	71.7(5.8)	60

likely found the solutions, but at considerably more cost). SPSA did well for 10^5 points, but otherwise failed to find a correct local optimum. DOT SQP QMC was consistently more accurate, more feasible, and much cheaper than the GA. SPSA was consistently

Table 4. Performance-based results for Problem 2. The initial point $d = \theta_0 = (10, 6)^t$.

No. Pts.	Reported Optimum Mean(σ)	System Analyses Mean(σ)	Optimizer Calls Mean(σ)	% Feas.
DOT (SQP), Quasi-MC Integrator				
10^1	—	—	—	—
10^2	35.89	4.42E+04	220.00	100
10^3	25.74	6.20E+04	30.00	100
10^4	—	—	—	—
10^5	25.24	2.68E+07	133.00	100
DOT (SQP), MC Integrator				
10^1	76.88(96.19)	2.76E+03(1.34E+03)	136.80(66.81)	20
10^2	151.99(127.36)	8.40E+03(3.00E+03)	41.00(15.00)	0
10^3	30.48(4.17)	2.92E+05(6.58E+04)	144.80(32.90)	50
10^4	28.40(2.95)	2.95E+06(1.57E+06)	146.67(78.53)	10
10^5	25.37(0.26)	1.24E+07(1.77E+06)	61.00(8.86)	30
SPSA, MC Integrator				
10^1	23.32(3.28) 410(44.1)	3.54E+03(4.56E+02)	81.6(6.9)	30
10^2	25.46(6.63)	2.36E+03(7.73E+02)	82.3(7.3)	50
10^3	21.84(5.64)	3.35E+04(6.45E+03)	89.8(10.1)	10
10^4	20.43(7.99)	7.85E+05(4.55E+04)	91.3(9.8)	0
10^5	22.96(5.55) 412(0.00)	6.66E+06(8.88E+04)	88.4(8.7)	20

cheaper than DOT SQP QMC, but less feasible, and correct only for 10^5 points. The best SQP result, for 10^4 points, was cheaper than the best SPSA result, for 10^5 points. Except for DOT SQP QMC, none of the algorithms did well on Problem 2. Overall, *DOT SQP QMC was more consistently accurate and feasible than the GA or SPSA, and the best SQP result was much cheaper than the best GA or SPSA result.*

For the other starting point $d = (10, 6)^t$ (Table 4), SPSA never found a correct solution, although it came close for 10^2 points, where it was cheaper than DOT SQP QMC. The fact that for more points SPSA never came close to a correct solution suggests that the good result for 10^2 points was an accident. Also note

the low percentages of feasible final points for SPSA. DOT SQP QMC also did not do well from this starting point, but it did get the correct result for 10^5 points.

6. Conclusions

Robust engineering design, being by definition design under uncertainty involving random variables, is inherently a stochastic optimization problem. Such problems have both deterministic and stochastic formulations, which can both be solved by either deterministic or nondeterministic algorithms. This work tested the reasonable presumption that nondeterministic algorithms (a genetic algorithm, Spall's simultaneous perturbation stochastic approximation algorithm) are better suited to robust design problems than standard deterministic optimization algorithms (method of feasible directions, sequential quadratic programming). The stochastic element of the test problems is an expectation integral $E[f(d, X)]$, which can be evaluated with varying accuracy, and whose approximation is itself a random variable. $E[f(d, X)]$ was approximated by (deterministic) quasi-Monte Carlo and (nondeterministic) Monte Carlo techniques, which agree to arbitrary precision (well beyond the optimization algorithm tolerances) for large enough samples. The comparison was between a fully deterministic standard optimization algorithm (MMFD or SQP with quasi-Monte Carlo integration) and a fully nondeterministic optimization algorithm (GA or SPSA with Monte Carlo integration) on several test problems with several starting points. With occasional exceptions, the overall conclusion is that a fully deterministic algorithm was consistently more accurate, more often feasible, and considerably more computationally efficient than both fully nondeterministic algorithms.

Acknowledgments

This work was supported in part by NSF Grants DMI-0422719 and DMI-0355391, and DOE Grant DE-FG02-06ER25720. The authors thank Sean Kugele for graciously assisting in some of the data collection.

References

[1] J. Sobieszczanski-Sobieski, R.T. Haftka, Multidisciplinary aerospace design optimization: survey and recent developments, *Structural and Multidisciplinary Optimization*, **14** (1997), 1–23.

[2] S. Burgee, L. Watson, *The promise (and reality) of multidisciplinary design optimization*, in *Large-Scale Optimization with Applications, Part II: Optimal Design and Control* L. T. Biegler, T. F. Coleman, A. R. Conn, F. N. Santosa (eds.), Springer-Verlag, New York (1997) 301–324.

[3] R. Unal, R. Lepsch, W. Engelund, D. Stanley, Approximation model building and multidisciplinary design optimization using response surface methods, in *Proc. 6th AIAA/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Part I*, (1996).

[4] J. Sacks, W. J. Welch, T. J. Mitchell, H. P. Wynn, Design and analysis of computer experiments, *Statistical Science*, **4** (1989), 409–435.

[5] E. R. Unger, M. G. Hutchison, M. Rais-Rohani, R. T. Haftka, B. Grossman, Variable-complexity multidisciplinary design of a transport wing, *International Journal of System Automation: Research and Applications (SARA)*, **2** (1992), 87–113.

[6] S.C. Kugele, M.W. Trosset, L.T. Watson, Numerical integration in statistical decision-theoretic methods for robust design optimization, *Structural and Multidisciplinary Optimization*, doi:10.1007/s00158-007-0189-0.

[7] W.J. Welch, J. Sacks, A system for quality improvement via computer experiments, *Communications in Statistics—Theory and Methods*, **20** (1991), 447–495.

[8] R.K. Roy, *A Primer on the Taguchi Method*, Society of Manufacturing, New York (1990).

[9] J.H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor (1975).

[10] P. Salaman, P. Sibani, R. Frost, *Facts, Conjectures, and Improvements for Simulated Annealing*, SIAM, Philadelphia (2002).

[11] A. Tettamanzi, M. Tomassini, *Soft Computing: Integrating Evolutionary, Neural, and Fuzzy Systems*, Springer-Verlag, New York (2001).

[12] J.C. Spall, Multivariate stochastic approximation using a simultaneous perturbation gradient approximation, *IEEE Transactions on Automatic Control*, **37** (1992), 332–341.

[13] M. S. Bazaraa, H. D. Sherali, C. M. Shetty, *Nonlinear Programming: Theory and Algorithms*, John Wiley & Sons, Inc., New York (1993).

[14] D. Haim, A. A. Giunta, M. M. Hozwarth, W. H. Mason, L. T. Watson, R. T. Haftka, Comparison of optimization software packages for an aircraft multidisciplinary design optimization problem, *Design Optimization*, **1** (1999), 9–23.

[15] M. T. McMahon, L. T. Watson, G. A. Soremekun, Z. Gürdal, R. T. Haftka, A Fortran 90 genetic algorithm module for composite laminate structure design, *Engineering with Computers*, **14** (1998), 260–273.