SIAM J. OPTIM.
Vol. 20, No. 1, pp. 172–191

BENCHMARKING DERIVATIVE-FREE OPTIMIZATION ALGORITHMS[*]

JORGE J. MORÉ[†] AND STEFAN M. WILD[‡]

Abstract. We propose *data profiles* as a tool for analyzing the performance of derivative-free optimization solvers when there are constraints on the computational budget. We use performance and data profiles, together with a convergence test that measures the decrease in function value, to analyze the performance of three solvers on sets of smooth, noisy, and piecewise-smooth problems. Our results provide estimates for the performance difference between these solvers, and show that on these problems, the model-based solver tested performs better than the two direct search solvers tested.

Key words. derivative-free optimization, benchmarking, performance evaluation, deterministic simulations, computational budget

AMS subject classifications. 90C56, 65Y20, 65K10

DOI. 10.1137/080724083

1. Introduction.

Derivative-free optimization has experienced a renewed interest over the past decade that has encouraged a new wave of theory and algorithms. While this research includes computational experiments that compare and explore the properties of these algorithms, there is no consensus on the benchmarking procedures that should be used to evaluate derivative-free algorithms.

We explore benchmarking procedures for derivative-free optimization algorithms when there is a limited computational budget. The focus of our work is the unconstrained optimization problem

$$\min \{ f(x) : x \in \mathbb{R}^n \}, \tag{1.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ may be noisy or nondifferentiable and, in particular, in the case where the evaluation of f is computationally expensive. These expensive optimization problems arise in science and engineering because evaluation of the function f often requires a complex deterministic simulation based on solving the equations (for example, nonlinear eigenvalue problems, ordinary or partial differential equations) that describe the underlying physical phenomena. The computational noise associated with these complex simulations means that obtaining derivatives is difficult and unreliable. Moreover, these simulations often rely on legacy or proprietary codes and hence must be treated as black-box functions, necessitating a derivative-free optimization algorithm.

Several comparisons have been made of derivative-free algorithms on noisy optimization problems that arise in applications. In particular, we mention [7, 10, 14, 17, 22]. The most ambitious work in this direction [7] is a comparison of six derivative-free optimization algorithms on two variations of a groundwater problem specified

[*]Received by the editors May 13, 2008; accepted for publication (in revised form) December 8, 2008; published electronically March 25, 2009.
http://www.siam.org/journals/siopt/20-1/72408.html

[†]Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439 (more@mcs.anl.gov). This work was supported by the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract DE-AC02-06CH11357.

[‡]Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439 (wild@mcs.anl.gov). This work was supported by a DOE Computational Science Graduate Fellowship under grant number DE-FG02-97ER25308.

by a simulator. In this work algorithms are compared by their trajectories (plot of the best function value against the number of evaluations) until the solver satisfies a convergence test based on the resolution of the simulator. The work in [7] also addresses *hidden constraints*, regions where the function does not return a proper value, a setting in which we have not yet applied the methodology presented here.

Benchmarking derivative-free algorithms on selected applications with trajectory plots provide useful information to users with related applications. In particular, users can find the solver that delivers the largest reduction within a given computational budget. However, the conclusions in these computational studies do not readily extend to other applications. Further, when testing larger sets of problems it becomes increasingly difficult to understand the overall performance of solvers using a single trajectory plot for each problem.

Most researchers have relied on a selection of problems from the CUTEr [9] collection of optimization problems for their work on testing and comparing derivative-free algorithms. Work in this direction includes [3, 14, 16, 18, 20]. The performance data gathered in these studies is the number of function evaluations required to satisfy a convergence test when there is a limit μ_f on the number of function evaluations. The convergence test is sometimes related to the accuracy of the current iterate as an approximation to a solution, while in other cases it is related to a parameter in the algorithm. For example, a typical convergence test for trust region methods [3, 18, 20] requires that the trust region radius be smaller than a given tolerance.

Users with expensive function evaluations are often interested in a convergence test that measures the decrease in function value. In section 2 we propose the convergence test

$$f(x_0) - f(x) \geq (1 - \tau)(f(x_0) - f_L), \tag{1.2}$$

where $\tau > 0$ is a tolerance, x_0 is the starting point for the problem, and f_L is computed for each problem as the smallest value of f obtained by any solver within a given number μ_f of function evaluations. This convergence test is well suited for derivative-free optimization because it is invariant to the affine transformation $f \mapsto \alpha f + \beta$ $(\alpha > 0)$ and measures the function value reduction $f(x_0) - f(x)$ achieved by x relative to the best possible reduction $f(x_0) - f_L$.

The convergence test (1.2) was used by Marazzi and Nocedal [16] but with f_L set to an accurate estimate of f at a local minimizer obtained by a derivative-based solver. In section 2 we show that setting f_L to an accurate estimate of f at a minimizer is not appropriate when the evaluation of f is expensive, since no solver may be able to satisfy (1.2) within the user's computational budget.

We use performance profiles [5] with the convergence test (1.2) to evaluate the performance of derivative-free solvers. Instead of using a fixed value of τ, we use $\tau = 10^{-k}$ with $k \in \{1, 3, 5, 7\}$ so that a user can evaluate solver performance for different levels of accuracy. These performance profiles are useful to users who need to choose a solver that provides a given reduction in function value within a limit of μ_f function evaluations.

To the authors' knowledge, previous work with performance profiles has not varied the limit μ_f on the number of function evaluations and has used large values for μ_f. The underlying assumption has been that the long-term behavior of the algorithm is of utmost importance. This assumption is not likely to hold, however, if the evaluation of f is expensive.

Performance profiles were designed to compare solvers and thus use a performance ratio instead of the number of function evaluations required to solve a problem. As a result, performance profiles do not provide the percentage of problems that can be solved (for a given tolerance τ) with a given number of function evaluations. This information is essential to users with expensive optimization problems and thus an interest in the short-term behavior of algorithms. On the other hand, the *data profiles* of section 2 have been designed to provide this information.

The remainder of this paper is devoted to demonstrating the use of performance and data profiles for benchmarking derivative-free optimization solvers. Section 2 reviews the use of performance profiles with the convergence test (1.2) and defines data profiles.

Section 3 provides a brief overview of the solvers selected to illustrate the benchmarking process: The Nelder-Mead NMSMAX code [13], the pattern-search APPSPACK code [11], and the model-based trust region NEWUOA code [20]. Since the emphasis of this paper is on the benchmarking process, no attempt was made to assemble a large collection of solvers. The selection of solvers was guided mainly by a desire to examine the performance of a representative subset of derivative-free solvers.

Section 4 describes the benchmark problems used in the computational experiments. We use a selection of problems from the CUTEr [9] collection for the basic set, but since the functions f that describe the optimization problem are invariably smooth, with at least two continuous derivatives, we augment this basic set with noisy and piecewise-smooth problems derived from this basic set. The choice of noisy problems was guided by a desire to mimic simulation-based optimization problems.

The benchmarking results in section 5 show that data and performance profiles provide complementary information that measures the strengths and weaknesses of optimization solvers as a function of the computational budget. Data profiles are useful, in particular, to assess the short-term behavior of the algorithms. The results obtained from the benchmark problems of section 4 show that the model-based solver NEWUOA performs better than the direct search solvers NMSMAX and APPSPACK even for noisy and piecewise-smooth problems. These results also provide estimates for the performance differences between these solvers.

Standard disclaimers [5] in benchmarking studies apply to the results in section 5. In particular, all solvers were tested with the default options, so results may change if these defaults are changed. In a similar vein, our results apply only to the current version of these solvers and this family of test problems, and may change with future versions of these solvers and other families of problems.

2. Benchmarking derivative-free optimization solvers.

Performance profiles, introduced by Dolan and Moré [5], have proved to be an important tool for benchmarking optimization solvers. Dolan and Moré define a benchmark in terms of a set $\mathcal{P}$ of benchmark problems, a set $\mathcal{S}$ of optimization solvers, and a convergence test $\mathcal{T}$. Once these components of a benchmark are defined, performance profiles can be used to compare the performance of the solvers. In this section we first propose a convergence test for derivative-free optimization solvers and then examine the relevance of performance profiles for optimization problems with expensive function evaluations.

2.1. Performance profiles.

Performance profiles are defined in terms of a performance measure $t_{p,s} > 0$ obtained for each $p \in \mathcal{P}$ and $s \in \mathcal{S}$. For example, this measure could be based on the amount of computing time or the number of function evaluations required to satisfy the convergence test. Larger values of $t_{p,s}$ indicate worse performance. For any pair (p, s) of problem p and solver s, the performance ratio is defined by

$$r_{p,s} = \frac{t_{p,s}}{\min\{t_{p,s} : s \in \mathcal{S}\}}.$$

Note that the best solver for a particular problem attains the lower bound $r_{p,s} = 1$. The convention $r_{p,s} = \infty$ is used when solver s fails to satisfy the convergence test on problem p.

The *performance profile* of a solver $s \in \mathcal{S}$ is defined as the fraction of problems where the performance ratio is at most α, that is,

$$\rho_s(\alpha) = \frac{1}{|\mathcal{P}|} \text{size}\{p \in \mathcal{P} : r_{p,s} \le \alpha\}, \tag{2.1}$$

where $|\mathcal{P}|$ denotes the cardinality of $\mathcal{P}$. Thus, a performance profile is the probability distribution for the ratio $r_{p,s}$. Performance profiles seek to capture how well the solver performs relative to the other solvers in $\mathcal{S}$ on the set of problems in $\mathcal{P}$. Note, in particular, that $\rho_s(1)$ is the fraction of problems for which solver $s \in \mathcal{S}$ performs the best and that for α sufficiently large, $\rho_s(\alpha)$ is the fraction of problems solved by $s \in \mathcal{S}$. In general, $\rho_s(\alpha)$ is the fraction of problems with a performance ratio $r_{p,s}$ bounded by α, and thus solvers with high values for $\rho_s(\alpha)$ are preferable.

Benchmarking gradient-based optimization solvers is reasonably straightforward once the convergence test is chosen. The convergence test is invariably based on the gradient, for example,

$$\|\nabla f(x)\| \le \tau \|\nabla f(x_0)\|$$

for some $\tau > 0$ and norm $\|\cdot\|$. This convergence test is augmented by a limit on the amount of computing time or the number of function evaluations. The latter requirement is needed to catch solvers that are not able to solve a given problem.

Benchmarking gradient-based solvers is usually done with a fixed choice of tolerance τ that yields reasonably accurate solutions on the benchmark problems. The underlying assumption is that the performance of the solvers will not change significantly with other choices of the tolerance and that, in any case, users tend to be interested in solvers that can deliver high-accuracy solutions. In derivative-free optimization, however, users are interested in both low-accuracy and high-accuracy solutions. In practical situations, when the evaluation of f is expensive, a low-accuracy solution is all that can be obtained within the user's computational budget. Moreover, in these situations, the accuracy of the data may warrant only a low-accuracy solution.

Benchmarking derivative-free solvers requires a convergence test that does not depend on evaluation of the gradient. We propose to use the convergence test

$$f(x) \le f_L + \tau (f(x_0) - f_L), \tag{2.2}$$

where $\tau > 0$ is a tolerance, x_0 is the starting point for the problem, and f_L is computed for each problem $p \in \mathcal{P}$ as the smallest value of f obtained by any solver within a given number μ_f of function evaluations. The convergence test (2.2) can also be written as

$$f(x_0) - f(x) \ge (1 - \tau)(f(x_0) - f_L),$$

and this shows that (2.2) requires that the reduction $f(x_0) - f(x)$ achieved by x be at least $1 - \tau$ times the best possible reduction $f(x_0) - f_L$.

The convergence test (2.2) was used by Elster and Neumaier [6] but with f_L set to an accurate estimate of f at a global minimizer. This test was also used by Marazzi and Nocedal [16] but with f_L set to an accurate estimate of f at a local minimizer obtained by a derivative-based solver. Setting f_L to an accurate estimate of f at a minimizer is not appropriate when the evaluation of f is expensive because no solver may be able to satisfy (2.2) within the user's computational budget. Even for problems with a cheap f, a derivative-free solver is not likely to achieve accuracy comparable to a derivative-based solver. On the other hand, if f_L is the smallest value of f obtained by any solver, then at least one solver will satisfy (2.2) for any $\tau \geq 0$.

An advantage of (2.2) is that it is invariant to the affine transformation $f \mapsto \alpha f + \beta$ where $\alpha > 0$. Hence, we can assume, for example, that $f_L = 0$ and $f(x_0) = 1$. There is no loss in generality in this assumption because derivative-free algorithms are invariant to the affine transformation $f \mapsto \alpha f + \beta$. Indeed, algorithms for gradient-based optimization (unconstrained and constrained) problems are also invariant to this affine transformation.

The tolerance $\tau \in [0, 1]$ in (2.2) represents the percentage decrease from the starting value $f(x_0)$. A value of $\tau = 0.1$ may represent a modest decrease, a reduction that is 90% of the total possible, while smaller values of τ correspond to larger decreases. As τ decreases, the accuracy of $f(x)$ as an approximation to f_L increases; the accuracy of x as an approximation to some minimizer depends on the growth of f in a neighborhood of the minimizer. As noted, users are interested in the performance of derivative-free solvers for both low-accuracy and high-accuracy solutions. A user's expectation of the decrease possible within their computational budget will vary from application to application.

The following new result relates the convergence test (2.2) to convergence results for gradient-based optimization solvers.

THEOREM 2.1. *Assume that $f : \mathbb{R}^n \mapsto \mathbb{R}$ is a strictly convex quadratic and that x^* is the unique minimizer of f. If $f_L = f(x^*)$, then $x \in \mathbb{R}^n$ satisfies the convergence test* (2.2) *if and only if*

$$\|\nabla f(x)\|_* \leq \tau^{1/2} \, \|\nabla f(x_0)\|_* \tag{2.3}$$

for the norm $\| \cdot \|_*$ defined by

$$\|v\|_* = \|G^{-\frac{1}{2}} \, v\|_2,$$

and G is the Hessian matrix of f.

Proof. Since f is a quadratic, G is the Hessian matrix of f, and x^* is the unique minimizer,

$$f(x) = f(x^*) + \tfrac{1}{2}(x - x^*)^T G (x - x^*).$$

Hence, the convergence test (2.2) holds if and only if

$$(x - x^*)^T G (x - x^*) \leq \tau (x_0 - x^*)^T G (x_0 - x^*),$$

which in terms of the square root $G^{\frac{1}{2}}$ is just

$$\|G^{\frac{1}{2}}(x - x^*)\|_2^2 \leq \tau \|G^{\frac{1}{2}}(x_0 - x^*)\|_2^2.$$

We obtain (2.3) by noting that since x^* is the minimizer of the quadratic f and G is the Hessian matrix, $\nabla f(x) = G(x - x^*)$. ☐

Other variations on Theorem 2.1 are of interest. For example, it is not difficult to show, by using the same proof techniques, that (2.2) is also equivalent to

$$\tfrac{1}{2}\|\nabla f(x)\|_*^2 \le \tau\,(f(x_0) - f(x^*)). \tag{2.4}$$

This inequality shows, in particular, that we can expect that the accuracy of x, as measured by the gradient norm $\|\nabla f(x)\|_*$, to increase with the square root of $f(x_0) - f(x^*)$.

Similar estimates hold for the error in x because $\nabla f(x) = G(x - x^*)$. Thus, in view of (2.3), the convergence test (2.2) is equivalent to

$$\|x - x^*\|_\diamond \le \tau^{1/2}\,\|x_0 - x^*\|_\diamond,$$

where the norm $\|\cdot\|_\diamond$ is defined by

$$\|v\|_\diamond = \|G^{\frac{1}{2}} v\|_2.$$

In this case the accuracy of x in the $\|\cdot\|_\diamond$ norm increases with the distance of x_0 from x^* in the $\|\cdot\|_\diamond$ norm.

We now explore an extension of Theorem 2.1 to nonlinear functions that is valid for an arbitrary starting point x_0. The following result shows that the convergence test (2.2) is (asymptotically) the same as the convergence test (2.4).

LEMMA 2.2. *If $f : \mathbb{R}^n \mapsto \mathbb{R}$ is twice continuously differentiable in a neighborhood of a minimizer x^* with $\nabla^2 f(x^*)$ positive definite, then*

$$\lim_{x\to x^*} \frac{f(x) - f(x^*)}{\|\nabla f(x)\|_*^2} = \frac{1}{2}, \tag{2.5}$$

where the norm $\|\cdot\|_*$ is defined in Theorem 2.1 *and $G = \nabla^2 f(x^*)$.*

Proof. We first prove that

$$\lim_{x\to x^*} \frac{\|\nabla^2 f(x^*)^{1/2}(x - x^*)\|}{\|\nabla f(x)\|_*} = 1. \tag{2.6}$$

This result can be established by noting that since $\nabla^2 f$ is continuous at x^* and $\nabla f(x^*) = 0$,

$$\nabla f(x) = \nabla^2 f(x^*)(x - x^*) + r_1(x), \qquad r_1(x) = o(\|x - x^*\|).$$

If λ_1 is the smallest eigenvalue of $\nabla^2 f(x^*)$, then this relationship implies, in particular, that

$$\|\nabla f(x)\|_* \ge \tfrac{1}{2}\lambda_1^{1/2}\|x - x^*\| \tag{2.7}$$

for all x near x^*. This inequality and the previous relationship prove (2.6). We can now complete the proof by noting that since $\nabla^2 f$ is continuous at x^* and $\nabla f(x^*) = 0$,

$$f(x) = f(x^*) + \tfrac{1}{2}\|\nabla^2 f(x^*)^{1/2}(x - x^*)\|^2 + r_2(x), \qquad r_2(x) = o(\|x - x^*\|^2).$$

This relationship, together with (2.6) and (2.7), completes the proof. ☐

Lemma 2.2 shows that there is a neighborhood $N(x^*)$ of x^* such that if $x \in N(x^*)$ satisfies the convergence test (2.2) with $f_L = f(x^*)$, then

$$\|\nabla f(x)\|_* \le \gamma\,\tau^{1/2}\,(f(x_0) - f(x^*))^{1/2}, \tag{2.8}$$

where the constant γ is a slight overestimate of $2^{1/2}$. Conversely, if γ is a slight underestimate of $2^{1/2}$, then (2.8) implies that (2.2) holds in some neighborhood of x^*. Thus, in this sense, the gradient test (2.8) is asymptotically equivalent to (2.2) for smooth functions.

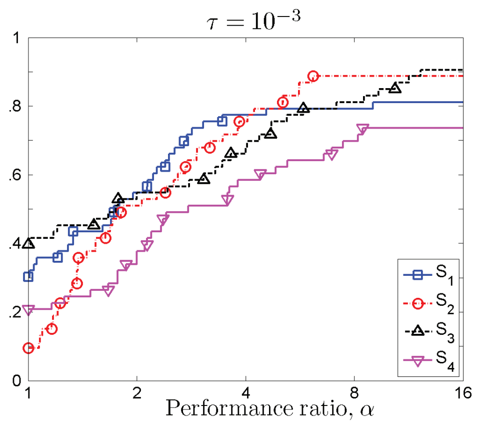

FIG. 2.1. *Sample performance profile $\rho_s(\alpha)$ (logarithmic scale) for derivative-free solvers.*

2.2. Data profiles. We can use performance profiles with the convergence test (2.2) to benchmark optimization solvers for problems with expensive function evaluations. In this case the performance measure $t_{p,s}$ is the number of function evaluations because this is assumed to be the dominant cost per iteration. Performance profiles provide an accurate view of the relative performance of solvers within a given number μ_f of function evaluations. Performance profiles do not, however, provide sufficient information for a user with an expensive optimization problem.

Figure 2.1 shows a typical performance profile for derivative-free optimization solvers with the convergence test (2.2) and $\tau = 10^{-3}$. Users generally are interested in the best solver, and for these problems and level of accuracy, solver S_3 has the best performance. However, it is also important to pay attention to the performance difference between solvers. For example, consider the performance profiles ρ_1 and ρ_4 at a performance ratio of $\alpha = 2$, $\rho_1(2) \approx 55\%$, and $\rho_4(2) \approx 35\%$. These profiles show that solver S_4 requires more than twice the number of function evaluations as solver S_1 on roughly 20% of the problems. This is a significant difference in performance.

The performance profiles in Figure 2.1 provide an accurate view of the performance of derivative-free solvers for $\tau = 10^{-3}$. However, these results were obtained with a limit of $\mu_f = 1300$ function evaluations and thus are not directly relevant to a user for which this limit exceeds their computational budget.

Users with expensive optimization problems are often interested in the performance of solvers as a function of the number of functions evaluations. In other words, these users are interested in *the percentage of problems that can be solved (for a given tolerance τ) with κ function evaluations.* We can obtain this information by letting $t_{p,s}$ be the number of function evaluations required to satisfy (2.2) for a given tolerance τ, since then

$$d_s(\alpha) = \frac{1}{|\mathcal{P}|}\text{size}\{p \in \mathcal{P} : t_{p,s} \leq \alpha\}$$

is the percentage of problems that can be solved with α function evaluations. As usual, there is a limit μ_f on the total number of function evaluations, and $t_{p,s} = \infty$ if the convergence test (2.2) is not satisfied after μ_f evaluations.

Griffin and Kolda [12] were also interested in performance in terms of the number of functions evaluations and used plots of the total number of solved problems as a function of the number of (penalty) function evaluations to evaluate performance. They did not investigate how results changed if the convergence test was changed; their main concern was to evaluate the performance of their algorithm with respect to the penalty function.

This definition of d_s is independent of the number of variables in the problem $p \in \mathcal{P}$. This is not realistic because, in our experience, the number of function evaluations needed to satisfy a given convergence test is likely to grow as the number of variables increases. We thus define the *data profile* of a solver $s \in \mathcal{S}$ by

$$d_s(\alpha) = \frac{1}{|\mathcal{P}|}\text{size}\left\{p \in \mathcal{P} : \frac{t_{p,s}}{n_p + 1} \le \alpha\right\}, \tag{2.9}$$

where n_p is the number of variables in $p \in \mathcal{P}$. We refer to a plot of (2.9) as a data profile to acknowledge that its application is more general than the one used here and that our choice of scaling is for illustration only. For example, we note that the authors in [1] expect performance of stochastic global optimization algorithms to grow faster than linear in the dimension.

With this scaling, the unit of cost is $n_p + 1$ function evaluations. This is a convenient unit that can be easily translated into function evaluations. Another advantage of this unit of cost is that $d_s(\kappa)$ can then be interpreted as the percentage of problems that can be solved with the equivalent of κ *simplex gradient estimates*, $n_p + 1$ referring to the number of evaluations needed to compute a one-sided finite-difference estimate of the gradient.

Performance profiles (2.1) and data profiles (2.9) are cumulative distribution functions, and thus monotone increasing, step functions with a range in $[0, 1]$. However, performance profiles compare different solvers, while data profiles display the raw data. In particular, performance profiles do not provide the number of function evaluations required to solve any of the problems. Also note that the data profile for a given solver $s \in \mathcal{S}$ is independent of other solvers; this is not the case for performance profiles.

Data profiles are useful to users with a specific computational budget who need to choose a solver that is likely to reach a given reduction in function value. The user needs to express the computational budget in terms of simplex gradients and examine the values of the data profile d_s for all the solvers. For example, if the user has a budget of 50 simplex gradients, then the data profiles in Figure 2.2 show that solver S_3 solves 90% of the problems at this level of accuracy. This information is not available from the performance profiles in Figure 2.1.

We illustrate the differences between performance and data profiles with a synthetic case involving two solvers. Assume that solver S_1 requires k_1 simplex gradients to solve each of the first n_1 problems, but fails to solve the remaining n_2 problems. Similarly, assume that solver S_2 fails to solve the first n_1 problems, but solves each of the remaining n_2 problems with k_2 simplex gradients. Finally, assume that $n_1 < n_2$, and that $k_1 < k_2$. In this case,

$$\rho_1(\alpha) \equiv \frac{n_1}{n_1 + n_2}, \qquad \rho_2(\alpha) \equiv \frac{n_2}{n_1 + n_2},$$

for all $\alpha \ge 1$ if the maximum number of evaluations μ_f allows k_2 simplex gradients. Hence, $n_1 < n_2$ implies that $\rho_1 < \rho_2$, and thus solver S_2 is preferable. This is justifiable because S_2 solves more problems for all performance ratios. On the other

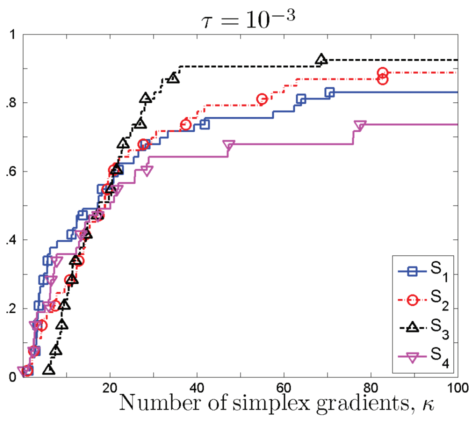

FIG. 2.2. *Sample data profile $d_s(\kappa)$ for derivative-free solvers.*

hand,

$$d_1(\alpha) = \begin{cases} 0, & \alpha \in [0, k_1) \\ \dfrac{n_1}{n_1 + n_2}, & \alpha \in [k_1, \infty) \end{cases} \qquad d_2(\alpha) = \begin{cases} 0, & \alpha \in [0, k_2) \\ \dfrac{n_2}{n_1 + n_2}, & \alpha \in [k_2, \infty) \end{cases}$$

In particular, $0 = d_2(k) < d_1(k)$ for all budgets of k simplex gradients where $k \in [k_1, k_2)$, and thus solver S_1 is preferable under these budget constraints. This choice is appropriate because S_2 is not able to solve any problems with less than k_2 simplex gradients.

This example illustrates an extreme case, but this can happen in practice. For example, the data profiles in Figure 2.2 show that solver S_2 outperforms S_1 with a computational budget of k simplex gradients where $k \in [20, 100]$, though the differences are small. On the other hand, the performance profiles in Figure 2.1 show that S_1 outperforms S_2.

One other connection between performance profiles and data profiles needs to be emphasized. The limiting value of $\rho_s(\alpha)$ as $\alpha \to \infty$ is the percentage of problems that can be solved with μ_f function evaluations. Thus,

$$d_s(\hat{\kappa}) = \lim_{\alpha \to \infty} \rho_s(\alpha), \tag{2.10}$$

where $\hat{\kappa}$ is the maximum number of simplex gradients performed in μ_f evaluations. Since the limiting value of ρ_s can be interpreted as the reliability of the solver, we see that (2.10) shows that the data profile d_s measures the reliability of the solver (for a given tolerance τ) as a function of the budget μ_f.

3. Derivative-free optimization solvers.

The selection of solvers $\mathcal{S}$ that we use to illustrate the benchmarking process was guided by a desire to examine the performance of a representative subset of derivative-free solvers, and thus we included both direct search and model-based algorithms. Similarly, our selection of solvers was not guided by their theoretical properties. No attempt was made to assemble a large collection of solvers, although we did consider more than a dozen different solvers. Users interested in the performance of other solvers (including SID-PSM [4] and UOBYQA [19]) can find additional results at www.mcs.anl.gov/˜more/dfo. We note that some solvers were not tested because they require additional parameters outside the scope of this investigation, such as the requirement of bounds by imfil [8, 15].

We considered only solvers that are designed to solve unconstrained optimization problems using only function values, and with an implementation that is both serial and deterministic. We used an implementation of the Nelder–Mead method because this method is popular among application scientists. We also present results for the APPSPACK pattern search method because, in a comparison of six derivative-free methods, this code performed well in the benchmarking [7] of a groundwater problem. We used the model-based trust region code NEWUOA because this code performed well in a recent comparison [18] of model-based methods.

The NMSMAX code is an implementation of the Nelder–Mead method and is available from the Matrix Computation Toolbox [13]. Other implementations of the Nelder–Mead method exist, but this code performs well and has a reasonable default for the size of the initial simplex. All variations on the Nelder–Mead method update an initial simplex defined by $n+1$ points via a sequence of reflections, expansions, and contractions. Not all of the Nelder–Mead codes that we examined, however, allow the size of the initial simplex to be specified in the calling sequence. The NMSMAX code requires an initial starting point x_0, a limit on the number of function evaluations, and the choice of a starting simplex. The user can choose either a regular simplex or a right-angled simplex with sides along the coordinate axes. We used the right-angled simplex with the default value of

$$\Delta_0 = \max\{1, \|x_0\|_\infty\} \tag{3.1}$$

for the length of the sides. This default value performs well in our testing. The right-angled simplex was chosen to conform with the default initializations of the two other solvers.

The APPSPACK code [11] is an asynchronous parallel pattern search method designed for problems characterized by expensive function evaluations. The code can be run in serial mode, and this is the mode used in our computational experiments. This code requires an initial starting point x_0, a limit on the number of function evaluations, the choice of scaling for the starting pattern, and an initial step size. We used unit scaling with an initial step size Δ_0 defined by (3.1) so that the starting pattern was defined by the right-angled simplex with sides of length Δ_0.

The model-based trust region code NEWUOA [20, 21] uses a quadratic model obtained by interpolation of function values at a subset of m previous trial points; the geometry of these points is monitored and improved if necessary. We used $m = 2n+1$ as recommended by Powell [20]. The NEWUOA code requires an initial starting point x_0, a limit on the number of function evaluations, and the initial trust region radius. We used Δ_0 as in (3.1) for the initial trust region radius.

Our choice of initial settings ensures that all codes are given the same initial information. As a result, both NMSMAX and NEWUOA evaluate the function at the vertices of the right-angled simplex with sides of length Δ_0. The APPSPACK code, however, moves off this initial pattern as soon as a lower function value is obtained.

We effectively set all termination parameters to zero so that all codes terminate only when the limit on the number of function evaluations is exceeded. In a few cases the codes terminate early. This situation happens, for example, if the trust region radius (size of the simplex or pattern) is driven to zero. Since APPSPACK requires a strictly positive termination parameter for the final pattern size, we used 10^{-20} for this parameter.

TABLE 4.1
Distribution of problem dimensions.

n_p	2	3	4	5	6	7	8	9	10	11	12
Number of problems	5	6	5	4	4	5	6	5	4	4	5

4. Benchmark problems.

The benchmark problems we have selected highlight some of the properties of derivative-free solvers as they face different classes of optimization problems. We made no attempt to define a definitive set of benchmark problems, but these benchmark problems could serve as a starting point for further investigations. This test set is easily available, widely used, and allows us easily examine different types of problems.

Our benchmark set comprises 22 of the nonlinear least squares functions defined in the CUTEr [9] collection. Each function is defined by m components $f_1, \ldots, f_m$ of n variables and a standard starting point x_s.

The problems in the benchmark set $\mathcal{P}$ are defined by a vector (k_p, n_p, m_p, s_p) of integers. The integer k_p is a reference number for the underlying CUTEr function, n_p is the number of variables, m_p is the number of components, and $s_p \in \{0, 1\}$ defines the starting point via $x_0 = 10^{s_p} x_s$, where x_s is the standard starting point for this function. The use of $s_p = 1$ is helpful for testing solvers from a remote starting point because the standard starting point tends to be close to a solution for many of the problems.

The benchmark set $\mathcal{P}$ has 53 different problems. No problem is overrepresented in $\mathcal{P}$ in the sense that no function k_p appears more than six times. Moreover, no pair (k_p, n_p) appears more than twice. In all cases,

$$2 \le n_p \le 12, \quad 2 \le m_p \le 65, \qquad p = 1, \ldots, 53,$$

with $n_p \le m_p$. The distribution of the dimensions n_p among all 53 problems is shown in Table 4.1, the median dimension being 7.

Users interested in the precise specification of the benchmark problems in $\mathcal{P}$ will find the source code for evaluating the problems in $\mathcal{P}$ at www.mcs.anl.gov/˜more/dfo. This site also contains source code for obtaining the standard starting points x_s and, a file dfo.dat that provides the integers (k_p, n_p, m_p, s_p).

We use the benchmark set $\mathcal{P}$ defined above to specify benchmark sets for three problem classes: Smooth, piecewise smooth, and noisy problems. The *smooth* problems $\mathcal{P}_S$ are defined by

$$f(x) = \sum_{k=1}^{m} f_k(x)^2. \tag{4.1}$$

These functions are twice continuously differentiable on the level set associated with x_0. Only two functions $(k_p = 7, 16)$ have local minimizers that are not global minimizers, but the problems defined by these functions appear only three times in $\mathcal{P}_S$.

The second class of problems mimics simulations that are defined by an iterative process, for example, solving to a specified accuracy a differential equation where the differential equation or the data depends on several parameters. These simulations are not stochastic, but do tend to produce results that are generally considered noisy. We believe the noise in this type of simulation is better modeled by a function with both high-frequency and low-frequency oscillations. We thus defined the *noisy* problems

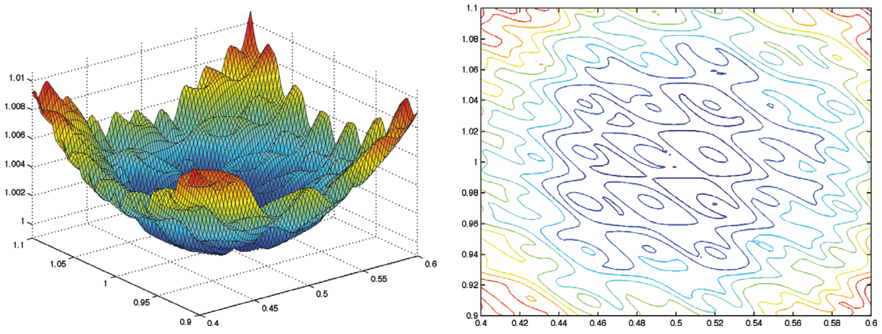

FIG. 4.1. *Plots of the noisy quadratic* (4.5) *on the box* $[0.4, 0.6] \times [0.9, 1.1]$. *Surface plots (left) and level sets (right) show the oscillatory nature of* f.

$\mathcal{P}_N$ by

$$f(x) = (1 + \varepsilon_f \phi(x)) \sum_{k=1}^{m} f_k(x)^2, \tag{4.2}$$

where ε_f is the relative noise level and the noise function $\phi : \mathbb{R}^n \mapsto [-1, 1]$ is defined in terms of the cubic Chebyshev polynomial T_3 by

$$\phi(x) = T_3(\phi_0(x)), \qquad T_3(\alpha) = \alpha(4\alpha^2 - 3), \tag{4.3}$$

where

$$\phi_0(x) = 0.9 \sin(100\|x\|_1) \cos(100\|x\|_\infty) + 0.1 \cos(\|x\|_2). \tag{4.4}$$

The function ϕ_0 defined by (4.4) is continuous and piecewise continuously differentiable with $2^n n!$ regions where ϕ_0 is continuously differentiable. The composition of ϕ_0 with T_3 eliminates the periodicity properties of ϕ_0 and adds stationary points to ϕ at any point where ϕ_0 coincides with the stationary points $(\pm\frac{1}{2})$ of T_3.

Figure 4.1 illustrates the properties of the noisy function (4.2) when the underlying smooth function $(\varepsilon_f = 0)$ is a quadratic function. In this case

$$f(x) = (1 + \tfrac{1}{2}\|x - x_0\|^2)(1 + \varepsilon_f \phi(x)), \tag{4.5}$$

where $x_0 = [\frac{1}{2}, 1]$, and noise level $\varepsilon_f = 10^{-3}$. The graph on the left shows f on the two-dimensional box around x_0 and sides of length $\frac{1}{2}$, while the graph on the right shows the contours of f. Both graphs show the oscillatory nature of f, and that f seems to have local minimizers near the global minimizer. Evaluation of f on a mesh shows that, as expected, the minimal value of f is 0.99906, that is, $1 - \varepsilon_f$ to high accuracy.

Our interest centers on smooth and noisy problems, but we also wanted to study the behavior of derivative-free solvers on piecewise-smooth problems. An advantage of the benchmark problems $\mathcal{P}$ is that a set of *piecewise-smooth* problems $\mathcal{P}_{PS}$ can be easily derived by setting

$$f(x) = \sum_{k=1}^{m} |f_k(x)|. \tag{4.6}$$

These problems are continuous, but the gradient does not exist when $f_k(x) = 0$ and $\operatorname{grad} f_k(x) \neq 0$ for some index k. They are twice continuously differentiable in the regions where all the f_k do not change sign. There is no guarantee that the problems in $\mathcal{P}_{PS}$ have a unique minimizer, even if (4.1) has a unique minimizer. However, we found that all minimizers were global for all but six functions and that these six functions had global minimizers only, if the variables were restricted to the positive orthant. Hence, for these six functions ($k_p = 8, 9, 13, 16, 17, 18$) the piecewise-smooth problems are defined by

$$f(x) = \sum_{k=1}^{m} |f_k(x_+)|, \tag{4.7}$$

where $x_+ = \max(x, 0)$. This function is piecewise-smooth and agrees with the function f defined by (4.6) for $x \geq 0$.

5. Computational experiments.

We now present the results of computational experiments with the performance measures introduced in section 2. We used the solver set $\mathcal{S}$ consisting of the three algorithms detailed in section 3 and the three problem sets $\mathcal{P}_S$, $\mathcal{P}_N$, and $\mathcal{P}_{PS}$ that correspond, respectively, to the smooth, noisy, and piecewise-smooth benchmark sets of section 4.

The computational results center on the short-term behavior of derivative-free algorithms. We decided to investigate the behavior of the algorithms with a limit of 100 simplex gradients. Since the problems in our benchmark sets have at most 12 variables, we set $\mu_f = 1300$ so that all solvers can use at least 100 simplex gradients.

Data was obtained by recording, for each problem and solver $s \in \mathcal{S}$, the function values generated by the solver at each trial point. All termination tolerances were set as described in section 3 so that solvers effectively terminate only when the limit μ_f on the number of function evaluations is exceeded. In the exceptional cases where the solver terminates early after $k < \mu_f$ function evaluations, we set all successive function values to $f(x_k)$. This data is then processed to obtain a history vector $h_s \in \mathbb{R}^{\mu_f}$ by setting

$$h_s(x_k) = \min \left\{ f(x_j) : 0 \leq j \leq k \right\},$$

so that $h_s(x_k)$ is the best function value produced by solver s after k function evaluations. Each solver produces one history vector for each problem, and these history vectors are gathered into a history array H, one column for each problem. For each problem, $p \in \mathcal{P}$, f_L was taken to be the best function value achieved by any solver within μ_f function evaluations, $f_L = \min_{s \in \mathcal{S}} h_s(x_{\mu_f})$.

We present the data profiles for $\tau = 10^{-k}$ with $k \in \{1, 3, 5, 7\}$ because we are interested in the short-term behavior of the algorithms as the accuracy level changes. We present performance profiles for only $\tau = 10^{-k}$ with $k \in \{1, 5\}$, but a comprehensive set of results is provided at www.mcs.anl.gov/˜more/dfo.

We comment only on the results for an accuracy level of $\tau = 10^{-5}$ and use the other plots to indicate how the results change as τ changes. This accuracy level is mild compared to classical convergence tests based on the gradient. We support this claim by noting that (2.8) implies that if x satisfies the convergence test (2.2) near a minimizer x^*, then

$$\|\nabla f(x)\|_* \leq 0.45 \cdot 10^{-2} \left(f(x_0) - f(x^*) \right)^{1/2}$$

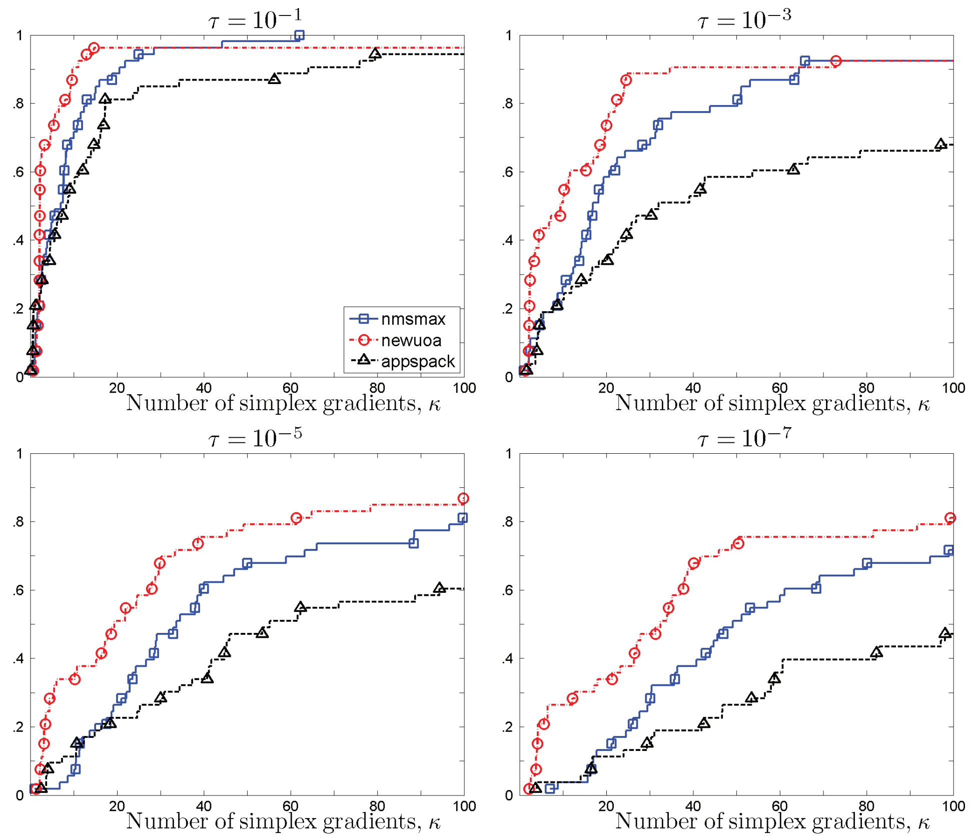

FIG. 5.1. *Data profiles $d_S(\kappa)$ for the smooth problems $\mathcal{P}_S$ show the percentage of problems solved as a function of a computational budget of simplex gradients.*

for $\tau = 10^{-5}$ and for the norm $\|\cdot\|_*$ defined in Theorem 2.1. If the problem is scaled so that $f(x^*) = 0$ and $f(x_0) = 1$, then

$$\|\nabla f(x)\|_* \leq 0.45 \cdot 10^{-2}.$$

This test is not comparable to a gradient test that uses an unscaled norm. It suggests, however, that for well-scaled problems, the accuracy level $\tau = 10^{-5}$ is mild compared to that of classical convergence tests.

5.1. Smooth problems. The data profiles in Figure 5.1 show that NEWUOA solves the largest percentage of problems for all sizes of the computational budget and levels of accuracy τ. This result is perhaps not surprising because NEWUOA is a model-based method based on a quadratic approximation of the function, and thus could be expected to perform well on smooth problems. However, the performance differences are noteworthy.

Performance differences between the solvers tend to be larger when the computational budget is small. For example, with a budget of 10 simplex gradients and $\tau = 10^{-5}$, NEWUOA solves almost 35% of the problems, while both NMSMAX and APPSPACK solve roughly 10% of the problems. Performance differences between NEWUOA and NMSMAX tend to be smaller for larger computational budgets. For example, with a budget of 100 simplex gradients, the performance difference between

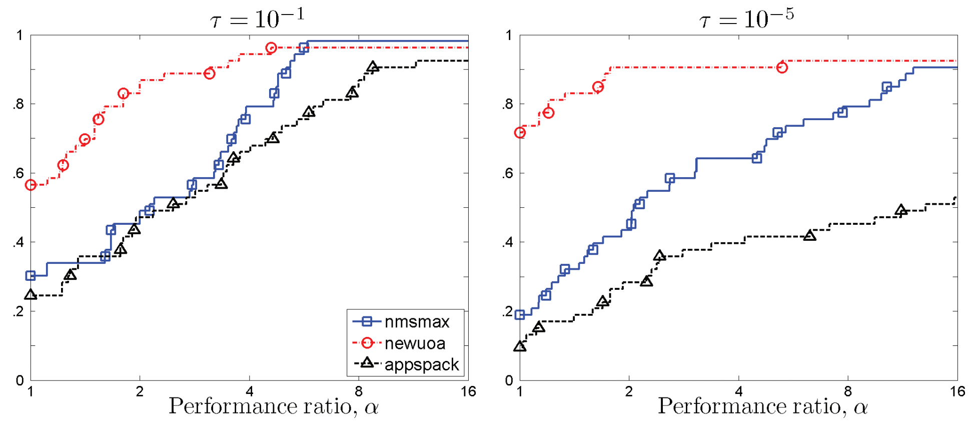

FIG. 5.2. *Performance profiles $\rho_s(\alpha)$ (logarithmic scale) for the smooth problems $\mathcal{P}_S$.*

NEWUOA and NMSMAX is less than 10%. On the other hand, the difference between NEWUOA and APPSPACK is more than 25%.

A benefit of the data profiles is that they can be useful for allocating a computational budget. For example, if a user is interested in getting an accuracy level of $\tau = 10^{-5}$ on at least 50% of problems, the data profiles show that NEWUOA, NMSMAX, and APPSPACK would require 20, 35, and 55 simplex gradients, respectively. This kind of information is not available from performance profiles because they rely on performance ratios.

The performance profiles in Figure 5.2 are for the smooth problems with a logarithmic scale. Performance differences are also of interest in this case. In particular, we note that both of these plots show that NEWUOA is the fastest solver in at least 55% of the problems, while NMSMAX and APPSPACK are each the fastest solvers on fewer than 30% of the problems.

Both plots in Figure 5.2 show that the performance difference between solvers decreases as the performance ratio increases. Since these figures are on a logarithmic scale, however, the decrease is slow. For example, both plots show a performance difference between NEWUOA and NMSMAX of at least 40% when the performance ratio is two. This implies that for at least 40% of the problems NMSMAX takes at least twice as many function evaluations to solve these problems. When $\tau = 10^{-5}$, the performance difference between NEWUOA and APPSPACK is larger, at least 50%.

5.2. Noisy problems.

We now present the computational results for the noisy problems $\mathcal{P}_N$ as defined in section 4. We used the noise level $\varepsilon_F = 10^{-3}$ with the nonstochastic noise function ϕ defined by (4.3, 4.4). We consider this level of noise to be about right for simulations controlled by iterative solvers because tolerances in these solvers are likely to be on the order of 10^{-3} or smaller. Smaller noise levels are also of interest. For example, a noise level of 10^{-7} is appropriate for single-precision computations.

Arguments for a nonstochastic noise function were presented in section 4, but here we add that a significant advantage of using a nonstochastic noise function in benchmarking is that this guarantees that the computational results are reproducible up to the precision of the computations. We also note that the results obtained with a noise function ϕ defined by a random number generator are similar to those

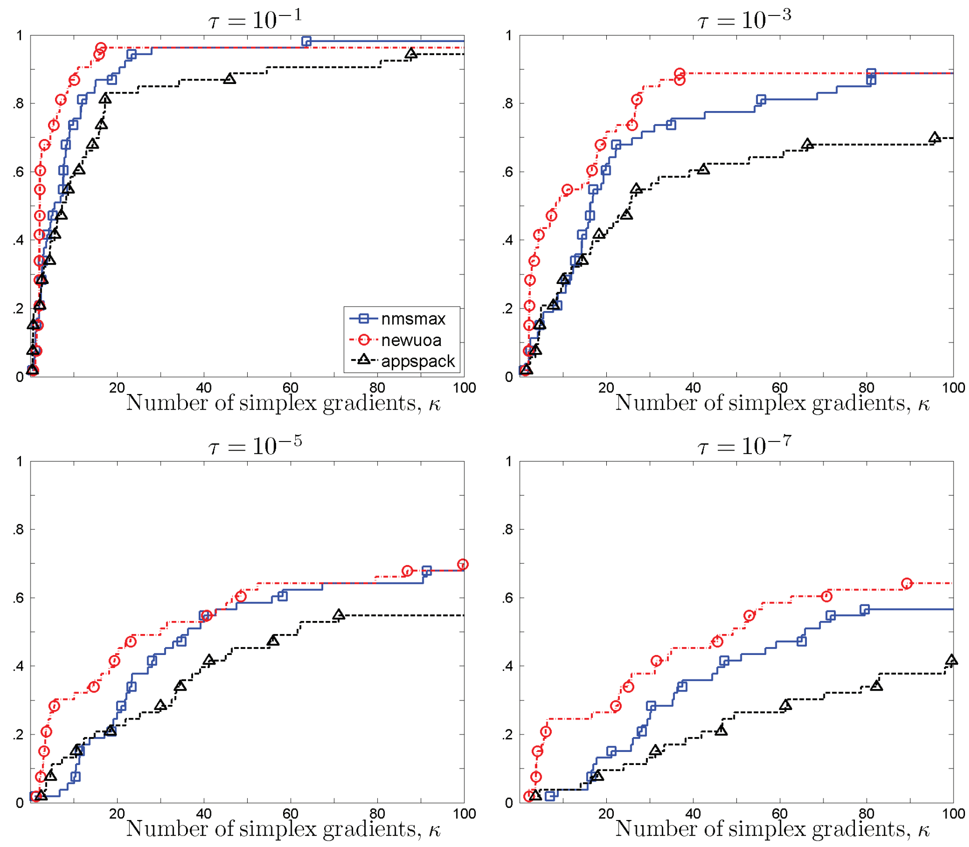

FIG. 5.3. *Data profiles $d_s(\kappa)$ for the noisy problems $\mathcal{P}_N$ show the percentage of problems solved as a function of a computational budget of simplex gradients.*

obtained by the ϕ defined by (4.3, 4.4); results for the stochastic case can be found at www.mcs.anl.gov/˜more/dfo.

The data profiles for the noisy problems, shown in Figure 5.3, are surprisingly similar to those obtained for the smooth problems. The degree of similarity between Figures 5.1 and 5.3 is much higher for small computational budgets and the smaller values of τ. This similarity is to be expected for direct search algorithms because the behavior of these algorithms depends only on logical comparisons between function values, and not on the actual function values. On the other hand, the behavior of NEWUOA is affected by noise because the model is determined by interpolating points and is hence sensitive to changes in the function values. Since NEWUOA depends on consistent function values, a performance drop can be expected for stochastic noise of magnitudes near a demanded accuracy level.

An interesting difference between the data profiles for the smooth and noisy problems is that solver performances for large computational budgets tend to be closer than in the smooth case. However, NEWUOA still manages to solve the largest percentage of problems for virtually all sizes of the computational budget and levels of accuracy τ.

Little similarity exists between the performance profiles for the noisy problems $\mathcal{P}_N$ when $\tau = 10^{-5}$, shown in Figure 5.4, and those for the smooth problems. In general

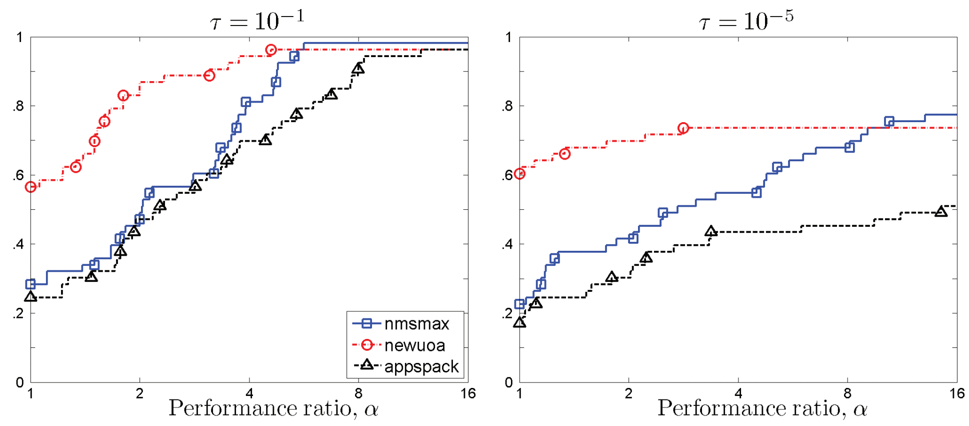

FIG. 5.4. *Performance profiles $\rho_s(\alpha)$ (logarithmic scale) for the noisy problems $\mathcal{P}_N$.*

these plots show that, as expected, noisy problems are harder to solve. For $\tau = 10^{-5}$, NEWUOA is the fastest solver on about 60% of the noisy problems, while it was the fastest solver on about 70% of the smooth problems. However, the performance differences between the solvers are about the same. In particular, both plots in Figure 5.4 show a performance difference between NEWUOA and NMSMAX of about 30% when the performance ratio is two. As we pointed out earlier, performance differences are an estimate of the gains that can be obtained when choosing a different solver.

5.3. Piecewise-smooth problems. The computational experiments for the piecewise-smooth problems $\mathcal{P}_{PS}$ measure how the solvers perform in the presence of nondifferentiable kinks. There is no guarantee of convergence for the tested methods in this case. We note that recent work has focused on relaxing the assumptions of differentiability [2].

The data profiles for the piecewise-smooth problems, shown in Figure 5.5, show that these problems are more difficult to solve than the noisy problems $\mathcal{P}_N$ and the smooth problems $\mathcal{P}_S$. In particular, we note that no solver is able to solve more than 40% of the problems with a computational budget of 100 simplex gradients and $\tau = 10^{-5}$. By contrast, almost 70% of the noisy problems and 90% of the smooth problems can be solved with this budget and level of accuracy. Differences in performance are also smaller for the piecewise smooth problems. NEWUOA solves the most problems in almost all cases, but the performance difference between NEWUOA and the other solvers is smaller than in the noisy or smooth problems.

Another interesting observation on the data profiles is that APPSPACK solves more problems than NMSMAX with $\tau = 10^{-5}$ for all sizes of the computational budget. This is in contrast to the results for smooth and noisy problem where NMSMAX solved more problems than APPSPACK.

The performance profiles for the piecewise-smooth problems $\mathcal{P}_{PS}$ appear in Figure 5.6. The results for $\tau = 10^{-5}$ show that NEWUOA, NMSMAX, and APPSPACK are the fastest solvers on roughly 50%, 30%, and 20% of the problems, respectively. This performance difference is maintained until the performance ratio is near $r = 2$. The same behavior can be seen in the performance profile with $\tau = 10^{-1}$, but now the initial difference in performance is larger, more than 40%. Also note that for $\tau = 10^{-5}$ NEWUOA either solves the problem quickly or does not solve the problem within μ_f evaluations. On the other hand, the reliability of both NMSMAX and APPSPACK in-

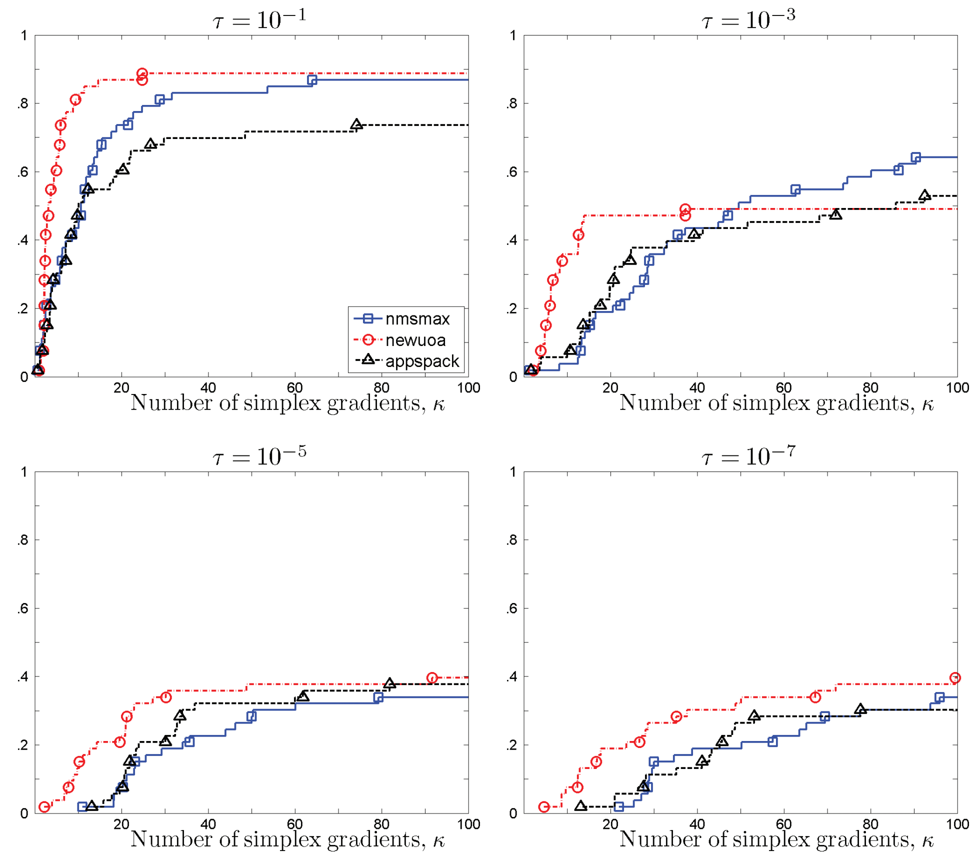

FIG. 5.5. *Data profiles $d_s(\kappa)$ for the piecewise-smooth problems $\mathcal{P}_{PS}$ show the percentage of problems solved as a function of a computational budget of simplex gradients.*

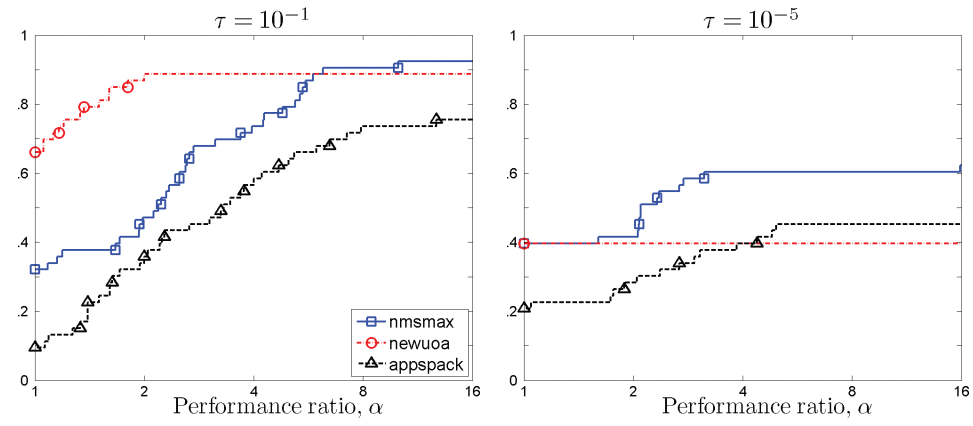

FIG. 5.6. *Performance profiles $\rho_s(\alpha)$ (logarithmic scale) for the piecewise-smooth problems $\mathcal{P}_{PS}$.*

creases with the performance ratio, and NMSMAX eventually solves more problems than NEWUOA.

Finally, note that the performance profiles with $\tau = 10^{-5}$ show that NMSMAX solves more problems than APPSPACK, while the data profiles in Figure 5.5 show that APPSPACK solves more problems than NMSMAX for a computational budget of k simplex gradients where $k \in [25, 100]$. As explained in section 2, this reversal of solver preference can happen when there is a constraint on the computational budget.

6. Concluding remarks.

Our interest in derivative-free methods is motivated in large part by the computationally expensive optimization problems that arise in DOE's SciDAC initiative. These applications give rise to the noisy optimization problems that have been the focus of this work.

We have used the convergence test (2.2) to define performance and data profiles for benchmarking unconstrained derivative-free optimization solvers. This convergence test relies only on the function values obtained by the solver and caters to users with an interest in the short-term behavior of the solver. Data profiles provide crucial information for users who are constrained by a computational budget and complement the measures of relative performance shown by performance plots.

Our computational experiments show that the performance of the three solvers considered varied from problem class to problem class, with the worst performance on the set of piecewise-smooth problems $\mathcal{P}_{PS}$. While NEWUOA generally outperformed the NMSMAX and APPSPACK implementations in our benchmarking environment, the latter two solvers may perform better in other environments. For example, our results did not take into account APPSPACK's ability to work in a parallel processing environment where concurrent function evaluations are possible. Similarly, since our test problems were unconstrained, our results do not readily extend to problems containing *hidden constraints*.

This work can be extended in several directions. For example, data profiles can also be used to benchmark solvers that use derivative information. In this setting we could use a gradient-based convergence test or the convergence test (2.2). Below we outline four other possible future research directions.

Performance on larger problems. The computational experiments in section 5 used problems with at most $n_p = 12$ variables. Performance of derivative-free solvers for larger problems is of interest, but this would require a different set of benchmark problems.

Performance on application problems. Our choice of noisy problems is a first step toward mimicking simulations that are defined by an iterative process, for example, solving a set of differential equations to a specified accuracy. We plan to validate this claim in future work. Performance of derivative-free solvers on other classes of simulations is also of interest.

Performance of other derivative-free solvers. As mentioned before, our emphasis is on the benchmarking process, and thus no attempt was made to assemble a large collection of solvers. Users interested in the performance of other solvers can find additional results at www.mcs.anl.gov/˜more/dfo. Results for additional solvers can be added easily.

Performance with respect to input and algorithmic parameters. Our computational experiments used default input and algorithmic parameters, but we are aware that performance can change for other choices.

Acknowledgments. The authors are grateful to Christine Shoemaker for useful conversations throughout the course of this work and to Josh Griffin and Luís Vicente for valuable comments on an earlier draft. Lastly, we are grateful for the developers of the freely available solvers we tested for providing us with source code and installation assistance: Genetha Gray and Tammy Kolda (APPSPACK [11]), Nick Higham

(MDSMAX and NMSMAX [13]), Tim Kelley (nelder and imfil [15]), Michael Powell (NEWUOA [20] and UOBYQA [19]), and Ana Custódio and Luís Vicente (SID-PSM [4]).

REFERENCES

[1] M. M. Ali, C. Khompatraporn, and Z. B. Zabinsky, *A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems*, J. Global Optim., 31 (2005), pp. 635–672.

[2] C. Audet and J. E. Dennis, *Analysis of generalized pattern searches*, SIAM J. Optim., 13 (2002), pp. 889–903.

[3] A. R. Conn, K. Scheinberg, and P. L. Toint, *A derivative free optimization algorithm in practice*, in Proceedings of 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, 1998.

[4] A. L. Custódio and L. N. Vicente, *Using sampling and simplex derivatives in pattern search methods*, SIAM J. Optim., 18 (2007), pp. 537–555.

[5] E. D. Dolan and J. J. Moré, *Benchmarking optimization software with performance profiles*, Math. Program., 91 (2002), pp. 201–213.

[6] C. Elster and A. Neumaier, *A grid algorithm for bound constrained optimization of noisy functions*, IMA J. Numer. Anal., 15 (1995), pp. 585–608.

[7] K. R. Fowler, J. P. Reese, C. E. Kees, J. E. Dennis, Jr., C. T. Kelley, C. T. Miller, C. Audet, A. J. Booker, G. Couture, R. W. Darwin, M. W. Farthing, D. E. Finkel, J. M. Gablonsky, G. Gray, and T. G. Kolda, *A comparison of derivative-free optimization methods for groundwater supply and hydraulic capture community problems*, Advances in Water Resources, 31 (2008), pp. 743–757.

[8] P. Gilmore and C. T. Kelley, *An implicit filtering algorithm for optimization of functions with many local minima*, SIAM J. Optim., 5 (1995), pp. 269–285.

[9] N. I. M. Gould, D. Orban, and P. L. Toint, *CUTEr and SifDec: A constrained and unconstrained testing environment, revisited*, ACM Trans. Math. Software, 29 (2003), pp. 373–394.

[10] G. A. Gray, T. G. Kolda, K. Sale, and M. M. Young, *Optimizing an empirical scoring function for transmembrane protein structure determination*, INFORMS J. Comput., 16 (2004), pp. 406–418.

[11] G. A. Gray and T. G. Kolda, *Algorithm* 856: *APPSPACK* 4.0: *Asynchronous parallel pattern search for derivative-free optimization*, ACM Trans. Math. Software, 32 (2006), pp. 485–507.

[12] J. D. Griffin and T. G. Kolda, *Nonlinearly-constrained optimization using asynchronous parallel generating set search*, Tech. Rep. SAND2007-3257, Sandia National Laboratories, Albuquerque, NM and Livermore, CA, May 2007, submitted.

[13] N. J. Higham, *The matrix computation toolbox*, www.ma.man.ac.uk/~higham/mctoolbox.

[14] P. D. Hough, T. G. Kolda, and V. J. Torczon, *Asynchronous parallel pattern search for nonlinear optimization*, SIAM J. Sci. Comput., 23 (2001), pp. 134–156.

[15] C. T. Kelley, *Users guide for imfil version* 0.5, available at www4.ncsu.edu/~ctk/imfil.html.

[16] M. Marazzi and J. Nocedal, *Wedge trust region methods for derivative free optimization*, Math. Program., 91 (2002), pp. 289–305.

[17] R. Oeuvray and M. Bierlaire, *A new derivative-free algorithm for the medical image registration problem*, Int. J. Modelling and Simulation, 27 (2007), pp. 115–124.

[18] R. Oeuvray, *Trust-region methods based on radial basis functions with application to biomedical imaging*, Ph.D. thesis, EPFL, Lausanne, Switzerland, 2005.

[19] M. J. D. Powell, *UOBYQA: Unconstrained optimization by quadratic approximation*, Math. Program., 92 (2002), pp. 555–582.

[20] M. J. D. Powell, *The NEWUOA software for unconstrained optimization without derivatives*, in Large Scale Nonlinear Optimization, G. Di Pillo and M. Roma, eds., Springer, Netherlands, 2006, pp. 255–297.

[21] M. J. D. Powell, *Developments of NEWUOA for unconstrained minimization without derivatives*, Preprint DAMTP 2007/NA05, University of Cambridge, Cambridge, England, 2007.

[22] R. G. Regis and C. A. Shoemaker, *A stochastic radial basis function method for the global optimization of expensive functions*, INFORMS J. Comput., 19 (2007), pp. 457–509.