Designing and Analyzing Computational Experiments for Global Optimization

Michael W. Trosset* Anthony D. Padula†

July 24, 2000

Abstract
We consider a variety of issues that arise when designing and analyzing computational experiments for global optimization. We describe a probability model for objective functions and a method for generating pseudorandom objective functions. We argue in favor of evaluating the performance of global optimization algorithms by measuring the depth of the objective function achieved with a fixed number of function evaluations. We emphasize the importance of replication in computational experiments and describe some useful statistical techniques for assimilating results. We illustrate our methods by performing a small study that compares two multistart strategies for global optimization.

Contents
1 Introduction 2
2 Generating Problems 2
  2.1 A Probability Model ................................................. 2
  2.2 Interpolation by Kriging ............................................. 3
  2.3 The Krigifier ............................................................. 3
  2.4 Illustrative Examples ................................................ 5
3 Measuring Performance and Analyzing Results 6
4 A Case Study 10

*Department of Mathematics, College of William & Mary, P.O. Box 8795, Williamsburg, VA 23185-8795. E-mail: trosset@math.wm.edu
†Department of Computational & Applied Mathematics, Rice University, Houston, TX 77005.
1 Introduction

It is widely accepted that the performance of algorithms for numerical optimization should be established in fact as well as in theory. Factual evidence includes the anecdotal experiences of users, but it should also include (as do other empirical sciences) the results of carefully designed experiments. Unfortunately, it is often not clear how to design and analyze meaningful computational experiments for numerical optimization. Part of the difficulty is the widespread perception that standard statistical methods are not germane. The purpose of the present work is to suggest some specific applications of statistical concepts to studies of how well numerical algorithms perform on global optimization problems.

Our suggestions are organized in two conceptual categories. In Section 2 we describe a stochastic method of generating objective functions. Our method allows a researcher to sample from a variety of infinite populations of functions. These populations can be customized so that the functions produced tend to exhibit certain characteristics. In Section 3 we describe some useful techniques for analyzing the results of global optimization experiments. First we assess some ways of measuring success when searching for a global minimizer. We argue that conventional criteria for measuring success when searching for a local minimizer are problematic when used in global optimization experiments. Instead, we recommend comparing objective function values after a fixed number of evaluations. We emphasize the concept of replication, which permits the researcher to draw valid inferences about the populations sampled, and describe some useful statistical procedures for doing so. We especially recommend constructing nonparametric probability density estimates from samples of putative minima. We illustrate our suggestions in Section 4, in which we report on a small study of two multistart strategies that we used with a pattern search algorithm to search for global minimizers. Although designed to be illustrative, our findings may be of independent interest.

2 Generating Problems

Individuals who study numerical optimization often recommend specific algorithms for specific applications. Typically, such recommendations are based partly on theory, partly on knowledge that the recommended algorithm has performed well on other, related applications. The latter rationale implicitly assumes that the relevant population of applications has been sufficiently well sampled to warrant making predictions about the new application in question. Is this usually the case? Computational experiments designed to assess the performance of algorithms for numerical optimization have traditionally relied on a small number of canonical test problems. This is particularly true for global optimization. There is an especially acute shortage of plausible objective functions of more than two variables that possess multiple local minimizers. To address this situation, we describe a method of generating an enormous variety of functions of an arbitrary number of variables with various numbers of local minimizers.

2.1 A Probability Model

Our approach to generating objective functions for global optimization was inspired by the literature on the design and analysis of computer experiments. Surveys of this literature include [13] and [5]. A computer experiment is a set of planned evaluations of a deterministic function $f : \mathbb{R}^p \rightarrow \mathbb{R}$, typically an expensive computer simulation of some physical phenomenon. Usually the goal of a computer experiment is to construct an inexpensive interpolation of $f$, $\hat{f} : \mathbb{R}^p \rightarrow \mathbb{R}$. 

2
Many of the ideas proposed for the design and analysis of computer experiments were motivated by assuming that \( f \) is a realization of a stochastic process. This assumption is a convenient fiction that has turned out to be quite helpful in suggesting plausible techniques. We use it here to develop a probability model for generating objective functions.

Let \( F \) denote a Gaussian random field, i.e., a collection of normal random variables \( F(x) \) indexed by \( x \in \mathbb{R}^p \), with mean function \( m(\cdot) \) and covariance function \( c(\cdot, \cdot) \). A realization of \( F \) is a function \( f: \mathbb{R}^p \to \mathbb{R} \). We propose to observe \( F \) at a finite number of design sites, obtaining \( y_i = f(x_i) \), then interpolate the observed values to obtain \( \hat{f}: \mathbb{R}^p \to \mathbb{R} \), a smoothed approximation of a realization of \( F \). As will be demonstrated in Section 2.4, by carefully choosing \( F \) and the number of points at which we observe \( F(x_i) \), we can obtain a remarkable variety of objective functions that can be customized to exhibit various characteristics. Furthermore, because these objective functions are generated by a probability model, computational experiments that employ them can be analyzed using conventional principles of statistical inference.

### 2.2 Interpolation by Kriging

Suppose that \( F \) is a random field with mean function \( m(x) = 0 \) and covariance function \( c(\cdot, \cdot) \). Suppose that \( f \) is a realization of \( F \) and that \( f(x_i) = y_i, i = 1, \ldots, n \). Given \( x \in \mathbb{R}^p \), we seek an estimator of \( f(x) \) that is of the form

\[
\hat{f}(x) = \sum_{i=1}^n b_i(x) f(x_i) = y^T b(x).
\]

We choose \( b(x) \) to minimize the mean squared error of prediction,

\[
E \left[ (\hat{f}(x) - F(x))^2 \right].
\]

This turns out to be a convex quadratic function of \( b \in \mathbb{R}^n \) with stationary equation \( Cb = d(x) \), where \( C = [c(x_i, x_j)] \in \mathbb{R}^{n \times n} \) and \( d(x) = [c(x_i, x)] \in \mathbb{R}^n \). If \( C \) is positive definite, then \( b(x) = C^{-1}d(x) \) and \( \hat{f}(x) = y^T C^{-1}d(x) \). This construction is called simple kriging. It is easily verified that \( \hat{f}(x_i) = y_i \).

Simple kriging can be extended to accommodate situations in which the mean function is an unknown constant (ordinary kriging) or an unknown linear combination of known functions (universal kriging), but we will rely on simple kriging. Kriging is widely used in geostatistics, spatial statistics, and the design and analysis of computer experiments. An accessible introduction, together with historical commentary and references, can be found in [1]. As discussed in [9], kriging is formally equivalent to interpolation by radial basis functions or thin-plate splines.

### 2.3 The Krigifier

This section describes the details of our procedure for generating pseudorandom objective functions. We refer to this procedure as the krigifier.\(^1\) A preliminary version of the krigifier was implemented in S-Plus and described by Trosset [18]; a more sophisticated version was implemented in C++ and studied by Padula [11]. The latter version (and accompanying documentation) can be obtained at http://www.cs.wm.edu/~va/software/.

The krigifier makes crucial use of the following fact:

**Lemma 1** If \( z \sim \text{Normal}(0, I) \), then \( Az \sim \text{Normal}(0, AA') \).

\(^1\)This name was suggested by Robert Michael Lewis.
Hence, to sample $y$ from a multivariate normal distribution with mean vector $0$ and positive definite covariance matrix $B$, one can do the following:

- **Multivariate Normal Sampling**
  
  To draw $y \sim \text{Normal}(0,B)$, first draw $z \sim \text{Normal}(0,I)$ and then do one of:

  - If $B$ is positive definite, then compute $B = LL'$, the Cholesky decomposition of $B$, and set $y = Lz$.
  - If $B$ is positive semidefinite, then compute $B = UU'$, the singular value decomposition of $B$, and set $y = UD^{1/2}U'z$.

  The krigifier can be organized into six conceptual components:

1. **Trend.** The user must specify a function $\text{trend}(\cdot)$. This might be a constant, e.g. $\text{trend}(x) = 0$, but it is often convenient to induce some underlying structure appropriate for nonlinear optimization, e.g. a convex quadratic function. Here is a simple way to generate pseudorandom convex quadratics of the form $\text{trend}(x) = x'Bx$:

- **Pseudorandom Convex Quadratic Trends**
  
  Given a covariance matrix $\bar{B}$,

  - Draw $x_1,\ldots,x_m \in \mathbb{R}^p$ from $\text{Normal}(0,\bar{B})$ and let $X = [x_1 \cdots x_m]'$.
  - Compute the sample covariance matrix $B = X'X/m$.

  If $m > p$, then $B$ is strictly positive definite with probability one. Notice that $B \xrightarrow{P} \bar{B}$ as $m \to \infty$, so smaller choices of $m$, e.g. $m = p + 1$, will result in greater sampling variability.

2. **Stochastic Process.** The user must specify a stationary Gaussian random field with mean zero and covariance function of the form

$$c(s,t) = \sigma^2 r(s,t).$$

(1)

The scalar $\sigma^2$ is the constant variance of the process at any point and the correlation function $r$ is selected from one of several common families. We have mainly experimented with correlation functions of the form

$$r(s,t) = \phi(\|s-t\|),$$

(2)

especially

$$\phi(u) = \exp(-\theta u^\alpha)$$

(3)

for $\alpha = 2$ and $\alpha = 1$. The former choice results in smooth ($C^\infty$) interpolations that seem better suited to generating nicely behaved objective functions; the latter choice results in jagged interpolations that seem better suited to simulating numerical noise. To specify a covariance function via equations (1)–(3), the user must specify the scalars $\alpha$, $\theta$, and $\sigma^2$.

3. **Design Sites.** The user must specify $n$, the number of sites (points in $[a,b]$) at which the stochastic process will be observed and subsequently interpolated. In principle, the locations of the sites can be chosen by any method whatsoever. We have typically drawn $x_1,\ldots,x_n \sim \text{Uniform}([a,b])$. 

4
4. **Observed Values.** To generate the observed values of the stochastic process at the sites \(x_1, \ldots, x_n\), let \(R = [r(x_i, x_j)]\) be the \(n \times n\) matrix of interpoint correlations and draw \(y\) from an \(n\)-variate normal distribution with covariance matrix \(\sigma^2 R\).

5. **Interpolation.** We interpolate the observed values by simple kriging. To do so, we compute \(\mathbf{v}\) by solving the square system of linear equations \(\mathbf{Rv} = \mathbf{y}\). Given \(x \in \mathbb{R}^p\), let

\[
\begin{bmatrix}
  r(x_1, x) \\
  \vdots \\
  r(x_n, x)
\end{bmatrix}.
\]

Then the interpolating function is

\[
\text{noise}(x) = \mathbf{v}' r(x).
\]

Notice that \(\text{noise}(x) = \sigma \cdot \text{noise}_1(x)\), where \(\text{noise}_1\) does not depend on \(\sigma\). Thus, \(\sigma\) determines the scale of the noise.

6. **Additive Noise.** The pseudorandom objective function is

\[
\hat{f}(x) = \text{trend}(x) + \text{noise}(x).
\]

Writing \(\hat{f}(x) = \text{trend}(x) + \sigma \cdot \text{noise}_1(x)\), we see that \(\sigma\) mediates the signal-to-noise ratio of \text{trend} to \text{noise}_1.

### 2.4 Illustrative Examples

We conclude our description of pseudorandom function generation by presenting some 1-dimensional examples. For simplicity, we restrict attention to functions of the form \(\hat{f}(x) = \text{noise}(x)\), in which case \(\sigma\) only affects the scaling of the function values. We define the covariance function by equations (1)–(3), with \(\sigma = 10\) in (1) and \(\alpha = 2\) in (3). This allows us to concentrate on the univariate correlation parameter, \(\theta\), in (3) and the number of design sites, \(n\). In our experience, it usually requires some trial and error to find values of these parameters that are suited to the project in question.

Our examples were constructed using \([a, b] = [0, 10] \subset \mathbb{R}\). Because the correlation between \(F(s)\) and \(F(t)\) depends on both \(\theta\) and \(\|s - t\|\), the choice of domain usually affects the choice of \(\theta\). We used \(\theta = 1, 5, 25\) and, for each value of \(\theta\), we generated pseudorandom functions using \(n = 20, 40, 80\) design sites. The nine functions that resulted are plotted in Figure 1, in which several instructive tendencies can be discerned.

First, for a fixed number of design sites, increasing \(\theta\) tends to result in \text{noise} functions with more oscillations. This occurs because increasing \(\theta\) decreases the correlation between \(F(s)\) and \(F(t)\) for \(\|s - t\|\) fixed. Thus, as \(\theta \to \infty\), the stochastic process begins to resemble white noise. For any finite set of design sites, \(\theta\) can be chosen sufficiently large that function values at any two sites are essentially unrelated. Once this is the case, further increasing \(\theta\) will have negligible effect. Choosing a large value of \(\theta\) may be desirable if one is trying to simulate numerical noise; we suspect that most users will prefer to generate less challenging objective functions.

Second, for a fixed value of the correlation parameter \(\theta\), increasing the number of design sites often results in \text{noise} functions with more oscillations. Increasing \(n\) permits more oscillations because there are more pseudorandom function values that must be interpolated. However, using a large number of design sites does not guarantee a large number of oscillations. Suppose that \(n\)
Figure 1: Nine pseudorandom functions of a single variable, $x \in [0, 10]$. Each function is of the form $f(x) = \text{noise}(x)$, with covariance function defined by equations (1)–(3) for $\alpha = 2$ and $\sigma = 10$. The first row was generated using $\theta = 1$, the second row using $\theta = 5$, and the third row using $\theta = 25$. The first column was generated using $n = 20$ design sites, the second column using $n = 40$ design sites, and the third column using $n = 80$ design sites.

is large, resulting in design sites that are tightly packed. If $s$ and $t$ are nearby design sites and $\theta$ is small enough that $F(s)$ and $F(t)$ are highly correlated, then the pseudorandom function values generated for these sites will be closely related—not likely to produce high-frequency oscillations when interpolated. This phenomenon is illustrated in the upper row of Figure 1. For $\theta = 1$, the functions generated using $n = 20, 40, 80$ design sites are surprisingly similar.

Even from these simple examples, it should be apparent that the inputs to the krigifier may interact in complicated ways, producing results that are not easily anticipated. We repeat our warning: some trial and error is usually required before the krigifier will produce functions that exhibit the features that one is seeking. Nevertheless, we believe that it is a powerful and flexible tool that can be used to perform an important task with a modest investment of time and energy.

3 Measuring Performance and Analyzing Results

The numerical optimization community has established a fairly conventional protocol for computational experiments that study convergence of iterative algorithms to local solutions. First the
researcher specifies suitable termination criteria; next the algorithms are run to termination; finally performance is assessed on the basis of (i) whether or not a solution was found and (ii) how much work was done to find it. We begin by arguing that this protocol is somewhat problematic for global optimization experiments.

Let us consider some typical convergence criteria. In their study of quasi-Newton methods, Dennis and Schmabel [2] organized their discussion of termination criteria around three heuristic questions:

1. “Have we solved the problem?”

2. “Have we grounded to a halt, either because the algorithm has converged or simply because it has stalled?”

3. “Have we exhausted our resources?”

If derivative information is available, then the first question is usually answered by ascertaining if the gradient of the objective function is sufficiently close to zero. In case the problem is badly scaled, a relative measure of magnitude is preferred. Let \(\text{typ} x_i\) denote the user’s estimates of typical magnitudes of the decision variables and let \(\text{typ} f\) denote the user’s estimate of a typical magnitude of the objective function. Then the termination criterion suggested by Dennis and Schmabel [2] is

\[
\max_{1 \leq i \leq p} \left| \frac{\nabla f(x_i^k)}{\text{typ} x_i} \right| \cdot \max \left( \frac{|x_i^k|}{\text{typ} x_i} \right) \leq \epsilon_1,
\]

where \(\epsilon_1\) is a user-specified threshold.

If derivative information is not available, then the first question must be answered indirectly, if at all. A popular termination criterion for the simplex algorithm of Nelder and Mead [10] is to compare the standard deviation of the function values at the current set of vertices to a user-specified threshold. Dolan, Lewis, and Torczon [4] show that the step control parameter for the generalized pattern search algorithms introduced by Torczon [16] is related to the gradient of the objective function and suggest comparing it to a user-specified threshold.

The second question is usually answered by ascertaining if the size of the current step is sufficiently close to zero. In the case of pattern search, the step control parameter can be used as a direct measure of step length, as well as an indirect measure of gradient magnitude. Dennis and Schmabel [2] suggest a relative measure of step length, viz.

\[
\max_{1 \leq i \leq p} \frac{|x_i^{k+1} - x_i^k|}{\text{typ} x_i} \leq \epsilon_2,
\]

where \(\epsilon_2\) is a user-specified threshold.

An alternative way of deciding if the algorithm has ground to a halt is to ascertain if the decrease in objective function values is sufficiently close to zero. However, if the objective function does not have much curvature, then the algorithm may terminate prematurely, when it is still taking steps of reasonable length. In consequence, this criterion does not have a good reputation in the numerical optimization community.

The first and second termination criteria are decisive in most studies of convergence to a local solution. When designing an experiment, the researcher usually anticipates that the algorithm will find a solution unless something goes terribly wrong. Computational resources are limited to ensure that the algorithm will terminate in worst-case scenarios, not to constrain its performance
in typical scenarios. For simplicity, one usually specifies an extremely generous maximum number of iterations.

How is performance measured in such studies? First, each attempt is declared to be a success or a failure according to whether it did or did not find a solution. Successes are usually associated with the first or second termination criteria; failures with the third termination criterion. Occasionally one notes an especially robust algorithm, i.e., an algorithm with an unusually high proportion of successes on the test problems considered. However, most of the proportions reported in the literature are so high that it is difficult to discriminate on this basis. More often, algorithms are compared on the basis of how much work was performed in finding a solution. Work can be measured in various ways, e.g., numbers of iterations, function and gradient evaluations, floating point operations, or CPU seconds.

We submit that the protocol we have just described is not well-suited for global optimization experiments. We begin by observing that the termination criteria involving gradients and step lengths are purely local. Although potentially useful for monitoring whether or not an algorithm has been trapped by a local minimizer, they cannot distinguish between global and nonglobal minimizers. Comparing values of the objective function is also problematic because many algorithms for global optimization do not produce a decreasing sequence of objective function values. Simulated annealing, for example, is predicated on a willingness to occasionally take steps that increase the value of the objective function.

Let us turn to the matter of how to measure performance. A global optimization algorithm is successful when it finds (approximately) a global minimizer. Assume, for simplicity, that \( f \) has a unique global minimizer \( x^* \). Then we might declare that an attempt is a success if it finds \( x^k \) for which \( \| x^k - x^* \| \) is sufficiently close to zero; alternatively, we might declare that an attempt is a success if it finds \( x^k \) for which \( f(x^k) - f(x^*) \) is sufficiently close to zero. In this context, because global optimization algorithms are often trapped by nonglobal minimizers, the proportion of successes is likely to be a quantity of compelling interest.

Unfortunately, because they assume knowledge of the global solution, our success criteria for global optimization do not lead to practical termination criteria. Furthermore, even if one wanted to study how much work is performed in finding a global solution, it may be difficult to do so. Finding global solutions is often extremely expensive, so that our third termination (exhausted resources) is often decisive. For this reason, we prefer a protocol for global optimization experiments that is dual to the conventional protocol for studying convergence to a local minimizer. Instead of measuring how much work was performed in finding a solution, we prefer to specify a budget that constrains the amount of work in a meaningful way, then measure the algorithm's progress toward a solution.

Budgets can be specified in various ways. Most of the studies that we have advised have involved direct search methods that do not use derivative information. In these studies, it was natural to limit the number of function evaluations available to each algorithm. The proportion of attempts that result in success is one important measure of progress, but it is a simplistic one that discards a great deal of information. We now consider some other measures of progress that we have found to be of immense diagnostic value.

The concept of replication is crucial to our understanding of progress. Too many computational experiments in numerical optimization perform a single attempt of each algorithm on each test problem. Occasionally this is appropriate, as when a problem is studied because of a certain pathology that is only exhibited when the algorithm starts from a certain initial point. We are highly critical, however, of the common tendency to declare that algorithm A solved problem P because a single attempt resulted in success. We submit that a thorough understanding of how A performs on P can be obtained only by starting A from a number of randomly generated initial points and drawing suitable inferences from the resulting data. One such inference is the estimated
probability that A will solve P, but we believe that other quantities are often more instructive.

Consider two global optimization algorithms. One tends to converge to nonglobal minimizers, the other finds nonstationary points that have smaller function values. From a global optimization perspective, it seems clear that the second algorithm outperforms the first. This is especially true if the second algorithm terminates because it exhausts its budget of resources, leaving open the possibility of further progress should additional resources be allocated. Thus, when evaluating the performance of a global optimization algorithm, we prefer measure performance according to the value of the objective function, not the location of the iterate. This reverses the perspective that prevails when studying convergence to a local solution.

Suppose that, for each attempt of algorithm A to solve problem P, we record the smallest objective function value discovered. We obtain a univariate sample (recording the corresponding $x^k$ results in a p-variate sample) that can be analyzed by familiar statistical techniques. One technique that we have found to be of enormous diagnostic value is construction of a nonparametric estimate of the probability density function from which the sample was drawn. The most familiar such estimator is the histogram; we prefer to use kernel methods. Software for both is widely available. See [15] and [14] for surveys of nonparametric density estimation.

As illustrated in Section 4, one can easily obtain a great deal of information by examining a plot of an estimated density function. The general tendency of the algorithm to return small (or large) function values is dramatically evident. So is the tendency of the algorithm to find a global solution. One can often detect convergence to local solutions. And plots of estimated density functions are of immense value when deciding what summary statistics to present and subject to further statistical analysis.

As for summary statistics and procedures for formal statistical inference (constructing confidence sets, testing statistical hypotheses), it is difficult to make general remarks. We envision that meaningful computational experiments for global optimization will comprise the following steps:

1. Specify the algorithms to be studied and the criteria for terminating them, e.g., a budget that constrains the computational resources available to the algorithms.

2. Specify the populations of problems on which the algorithms are to be tested.

3. Draw sample problems from the specified populations, e.g., by the methods described in Section 2.

4. Attempt to solve each problem with each algorithm a great many times, e.g., by starting each algorithm from each of a number of randomly generated initial points.

5. Record suitable measures of performance, e.g., the best function value obtained in each attempt. Perform exploratory analyses of (some of) these data, e.g., by constructing nonparametric estimates of the probability density functions.

6. Compute statistics that summarize how well each algorithm performed on each problem.

7. Draw inferences about how the algorithms compare with respect to the specified populations. For example, one might test the null hypothesis that algorithm A is no better than algorithm B on problems drawn from population $\mathcal{P}$.

This prescription allows tremendous flexibility. Statistics is a vast discipline and no one can comment on all of the potentially relevant procedures. We prefer to illustrate some possibilities by reporting a small study that we performed for this purpose. It is to this study that we now turn.
4  A Case Study

We illustrate the ideas introduced in Sections 2 and 3 by conducting a small study. Because this study is only illustrative, we leave numerous issues unexplored; nevertheless, we hope that our results are of independent interest.

Our study compares two multistart strategies that one might adopt when using a local search algorithm to try to find a global minimizer of \( f : \mathbb{R}^p \rightarrow \mathbb{R} \) in the rectangular feasible region \([a, b] \subset \mathbb{R}^p\). The local search algorithm that we use is a member of the class of generalized pattern search algorithms. These direct search algorithms were introduced by Torczon [16]; see [17, 7, 8] for elementary expositions. Specifically, we use a pattern search algorithm that searches from the current iterate in the positive and negative coordinate directions. The first trial step that produces simple decrease of the objective function is accepted and the step length is halved when no trial step produces decrease. Dolan [3], whose software we used, refers to this algorithm as compass search. Compass search manages constraints by a standard device: infeasible points are assigned infinite objective function values. Lewis and Torczon [6] have demonstrated that, for the case of bound constraints, compass search is globally convergent to a constrained stationary point.

A standard (and often effective) strategy for solving global optimization problems is to start a local search algorithm from a number of different points. However, multistart strategies can be implemented in different ways with potentially different consequences. For \( p = 2 \) or \( p = 4 \), a pseudorandom objective function \( f : [0, 10]^p \rightarrow \mathbb{R} \), and a total budget of \( V = 1000 \) function evaluations, we consider two distinct possibilities:

1. **Sequential Multistart Compass Search**

   Do until the budget is exhausted:
   
   (a) Draw \( x_0 \sim \text{Uniform}([0, 10]^p) \).

   (b) Run compass search from \( x_0 \) with initial step length \( \Delta_0 = 2 \) until convergence.

2. **Parallel Multistart Compass Search**

   (a) Create a Latin hypercube of \( N = 10 \) design sites in \([0, 10]^p\).

   (b) Run compass search from each design site with initial step length \( \Delta_0 = 2 \) until a budget of \( V_t = 99 \) is exhausted.

The sequential multistart strategy was conceived to place primary emphasis on finding local minimizers and secondary emphasis on finding basins that contain small function values. Regardless of expense, it will complete a search for a local minimizer before beginning another search. In contrast, the parallel multistart strategy was conceived to place primary emphasis on finding small function values. It is committed to perform \( N = 10 \) local searches, even if none of them is completed. Thus, it was intended that the sequential strategy would perform fewer, but more comprehensive, local searches than the parallel strategy. Notice that choices of \( V \) and \( N \) that realize this intent depend on \( p \), \( f \), and the local search algorithm. For example, if derivative information and a quasi-Newton algorithm were used, the sequential strategy might complete more than 10 local searches before exhausting the total budget.

Several technical details should be noted. Both strategies were implemented in serial, although the parallel strategy could exploit multiple processors if they were available. The choice of initial step length was based on recommendations by Parker [12]. Following Dolan [3], the convergence criterion for local search was \( \Delta_k < 2 \times 10^{-8} \).

Finally, Latin hypercubes are easily computed, randomly generated, space-filling experimental designs for rectangles. Given \([a, b] \subset \mathbb{R}^p \) and \( N \), one constructs a Latin hypercube as follows:
Latin Hypercube Design

(a) For $i = 1, \ldots, p$, partition $[a_i, b_i]$ into $N$ subintervals of length $(b_i - a_i)/N$. This partitions $[a, b]$ into $N^p$ rectangles, $R(j_1, \ldots, j_p)$, where $j_i$ identifies interval $i$ in $[a_i, b_i]$.

(b) For $i = 1, \ldots, p$, generate a random permutation $J_{i1}, \ldots, J_{iN}$ of $\{1, \ldots, N\}$.

(c) For $n = 1, \ldots, N$, draw $x_0 \sim \text{Uniform}(R(J_{in}, \ldots, J_{ln}))$.

Latin hypercubes are space-filling in the following sense: if the set of design sites is projected into any $[a_i, b_i]$, then each subinterval of $[a_i, b_i]$ contains exactly one projected design site. Some sense of such a design can be obtained by positioning $N = 8$ rooks on a chessboard in such a way that no rook can capture another.\(^2\) See [5] for more information about Latin hypercubes and other space-filling experimental designs.

Two populations of objective functions were considered, one for $p = 2$ and one for $p = 4$. These populations were sampled by the krigifier, as described in Section 2.3. Thus, each objective function was of the form $f_p(x) = \text{trend}_p(x) + \text{noise}_p(x)$. Each trend function was a pseudorandom convex quadratic function of the form $\text{trend}_p(x) = 100 + (x - u_p)'B_p(x - u_p)$, where $B_p$ was a sample covariance matrix obtained by drawing $p + 1$ observations from a $p$-variate normal distribution with mean vector zero and covariance matrix $\tilde{B}_p = \tau^2I_p$ and $u_p \sim \text{Uniform}(\{2.5, 7.5\}^p)$. We used $\tau^2 = 1$. Each noise function was obtained by first drawing design sites, $x_1, \ldots, x_n \sim \text{Uniform}(0, 10]^p$, then drawing a corresponding vector of noise function values from an $n$-variate normal distribution with covariance matrix $\sigma^2R$, then interpolating these values by simple kriging. The correlation matrix $R = [r(x_i, x_j)]$ was defined by equations (2) and (3) with $\alpha = 2$. We used $n = 50$ design sites, variance $\sigma^2 = 100$, and correlation parameter $\theta = 1$. Our choices of $(\tau, \sigma, n, \theta)$ were made after some preliminary experimentation revealed that these choices tended to produce $f_p$ with roughly 5–10 local minimizers. A typical $\hat{f}_2$ is plotted in Figure 2.

Three samples were collected from each pseudorandom objective function. Sequentialmultistart compass search (SMCS) and parallel multistart compass search (PMCS) were each run 1000 times and the best function value from each run was recorded. In addition, compass search was run to convergence ($\Delta_k < 2 \times 10^{-8}$) from each of 1000 initial points drawn from Uniform$(0, 10]^p$ and the best function value from each run was recorded. This auxiliary data set was used to determine the local minima of the objective function.

Exploratory analyses of the two samples (SMCS and PMCS) of best function values were conducted for several $\hat{f}_2$ and $\hat{f}_4$. Nonparametric probability density function estimates with rectangular kernels were constructed using the density function in the statistical programming language S-Plus. For the $\hat{f}_2$ objective function plotted in Figure 2, the density estimates so constructed are plotted in Figure 3. Corresponding estimates for an $\hat{f}_4$ objective function are plotted in Figure 4.

The estimated density functions plotted in Figures 3 and 4 should be interpreted in the same way that one would interpret a histogram, viz. the integral of the density function over any specified interval approximates the proportion of runs that the algorithm returned a best function value in that interval. The dotted vertical lines indicate the minima of the objective function in question. If multiple runs of the algorithm produce the same minimum, then the estimated density function will have a sharp peak at that value. Notice that these peaks are approximately centered at the minima, suggesting values slightly greater than and less than the true minima. The latter is usually an artifact of our estimation procedure, which centers a rectangular kernel at each value in the sample. This artifact could be avoided by left-justifying an asymmetric kernel at each sample point, but doing so would require writing specialized software.

\(^2\)This observation is due to Christopher Siefert.
Figure 2: A pseudorandom function \( \hat{f}_2 \) drawn from the population of 2-dimensional functions on which the sequential and parallel multistart compass search strategies were tested.

Let us first consider how SMCS and PMCS performed on the objective function \( \hat{f}_2 \) plotted in Figure 2. The first density function plotted in Figure 3 reveals that the overwhelming majority of the 1000 SMCS runs effectively converged to one of the three smallest minima of \( \hat{f}_2 \). Each of these minima was returned by a substantial proportion of the runs and the second smallest minimum was returned more frequently than either of the other two. In contrast, the second density function plotted in Figure 3 reveals that the overwhelming majority of the 1000 PMCS runs effectively converged to one of the two smallest minima of \( \hat{f}_2 \). The global minimum was returned much more frequently than the nonglobal minimum.

It seems evident that PMCS is better than SMCS at finding the global minimum of \( \hat{f}_2 \) plotted in Figure 2. The fact that the overwhelming majority of the PMCS runs effectively converged to a local minimum suggests that a budget of \( V = 99 \) function evaluations is adequate for compass search to find a local minimizer of \( \hat{f}_2 \). Actually, compass search usually required nearly twice that many function evaluations to satisfy the convergence criterion \( \Delta_k < 2 \times 10^{-8} \). Thus, PMCS is not quite converging in the same sense that SMCS is converging, but the resolution used to construct the density estimates in Figure 2 masks the difference. This difference is a critical concern when studying local convergence; we submit that it is of little consequence when studying global optimization. Indeed, we would argue that SMCS overemphasizes local convergence, squandering function evaluations that might better be used to perform additional searches for the basin that contains the global minimizer. We also note the potential benefits of drawing (by Latin hypercube
Figure 3: Distributions of best function values found by 1000 runs of SMCS and PMCS on the pseudorandom objective function \( f_2 \) plotted in Figure 2. The dotted vertical lines indicate the minima of \( f_2 \).

Now let us first consider how SMCS and PMCS performed on a pseudorandom objective function \( f_4 \). The increase in dimension means that compass search will require more function evaluations to converge for \( p = 4 \) than for \( p = 2 \). Thus, SMCS will tend to conduct fewer searches and PMCS will tend to converge less effectively. Each of these tendencies can be discerned in Figure 4.

The first density function plotted in Figure 4 reveals that the overwhelming majority of the 1000 SMCS runs effectively converged to one of the four smallest minima of \( f_4 \). Unfortunately, a large proportion of these runs converged to the fourth smallest minimum. In contrast, the second density function plotted in Figure 3 reveals that many of the 1000 PMCS runs did not converge to a local minimum, suggesting that \( V = 99 \) is a fairly stringent budget for compass search in \( p = 4 \) dimensions. However, most of the PMCS runs did return values that were reasonably near a local minimum. Compensating for its less effective convergence, PMCS exhibited a dramatic tendency to return smaller values of \( f_4 \) than did SMCS. In particular, PMCS found the basin that contains the global minimizer of \( f_4 \) far more frequently than did SMCS.
Figure 4: Distributions of best function values found by 1000 runs of SMCS and PMCS on a pseudorandom objective function \( f_4 \). The dotted vertical lines indicate the minima of \( f_4 \).

We hope that we have persuaded the reader that a great deal of useful information can be discerned in estimated density functions of the sort plotted in Figures 3 and 4. However, each of these figures corresponds to a single objective function, whereas inferences about algorithm performance should be based on a sample of objective functions drawn from one (or more) specified populations. Examining a set of density plots for each of a great many functions may be highly illuminating, but it will also be rather unwieldy. Even if a researcher is motivated to construct and examine so many plots, he or she will still need to find ways to summarize any general patterns that emerge. To illustrate, we performed the following experiment:

1. Draw 100 pseudorandom objective functions \( \hat{f}_p \) from each of the previously designated populations (\( p = 2 \) and \( p = 4 \)).

2. For each \( \hat{f}_p \):

   (a) Draw 1000 points from Uniform([0, 10]^{p_1}). Run compass search from each point, with initial step length \( \Delta_0 = 2 \), until \( \Delta_k < 2 \times 10^{-8} \). Record the number of local minimizers found.
(b) Perform 1000 runs of SMCS. Record the number of runs for which SMCS returned a value smaller than the second smallest minimum of \( \hat{f}_p \).

(c) Perform 1000 runs of PMCS. Record the number of runs for which PMCS returned a value smaller than the second smallest minimum of \( \hat{f}_p \).

The 200 pseudorandom objective functions that we generated exhibited various numbers of local minimizers, ranging from 1 local minimizer (for 2 of the functions of \( p = 2 \) variables and 1 of the functions of \( p = 4 \) variables) to 14 local minimizers (for 2 of the functions of 4 variables). For \( p = 2, 4 \) variables and \( k = 1, \ldots, 14 \) local minimizers, we observed the following numbers of objective functions:

<table>
<thead>
<tr>
<th># mins</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p = 2 )</td>
<td>2</td>
<td>0</td>
<td>5</td>
<td>13</td>
<td>16</td>
<td>21</td>
<td>22</td>
<td>10</td>
<td>8</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( p = 4 )</td>
<td>1</td>
<td>4</td>
<td>9</td>
<td>12</td>
<td>10</td>
<td>12</td>
<td>15</td>
<td>16</td>
<td>6</td>
<td>6</td>
<td>2</td>
<td>5</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

Examining these numbers allows us to describe more precisely the populations that we have chosen to study. In particular, these numbers suggest that our pseudorandom objective functions tend be challenging—but not impossibly difficult—functions on which to test global optimization algorithms, thereby strengthening the case for using such functions in computational experiments.

Also associated with each \( \hat{f}_p \) is a pair of numbers that record how many of the 1000 SMCS runs and how many of the 1000 PMCS runs were successful, where a run was considered successful if a function value less than the second smallest local minimum of \( \hat{f}_p \) was returned. Thus, our experiment produced two bivariate data sets, one for \( p = 2 \) and one for \( p = 4 \), each comprising 100 observations. Scatter diagrams of these data sets are displayed in Figure 5.

Each point in Figure 5 corresponds to a different \( \hat{f}_p \). The dotted diagonal lines indicate where a point would lie if SMCS and PMCS were equally successful for the corresponding \( \hat{f}_p \). Points above the diagonal correspond to \( \hat{f}_p \) for which PMCS was more successful than SMCS; points below the diagonal correspond to \( \hat{f}_p \) for which PMCS was less successful than SMCS. *Not one point falls below the diagonal.* These diagrams are so compelling that further analysis is superfluous, but we continue for the sake of illustration.

It is not difficult to construct summary statistics that convey the essence of Figure 5. For example, for each \( \hat{f}_p \) we might ask whether or not each algorithm was successful in at least 95 percent of its runs. Counting the number of \( \hat{f}_p \) for which each algorithm was so successful, we obtain the following contingency table:

<table>
<thead>
<tr>
<th></th>
<th>SMCS</th>
<th>PMCS</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p = 2 )</td>
<td>25</td>
<td>99</td>
</tr>
<tr>
<td>( p = 4 )</td>
<td>11</td>
<td>50</td>
</tr>
</tbody>
</table>

This table provides succinct testimony to the superiority of PMCS for the populations studied.

One might also endeavor to devise more exotic summary statistics that quantify the extent to which the points in Figure 5 lie above their respective diagonals. For each scatter diagram, let \((S_i, P_i)\) denote point \( i \). Ignoring the points for which \( S_i = P_i = 1000 \), let \( r_i = (P_i - S_i)/(1000 - S_i) \) and let \( r \) equal the mean of the \( r_i \). Then \( r = 0 \) if all of the points lie on the diagonal and \( r = 1 \) if all of the points lie on the horizontal line \( P = 1000 \). In fact, \( r = .992 \) for \( p = 2 \) and \( r = .734 \) for \( p = 4 \). These statistics document our strong visual impression that most of the points in Figure 5 lie substantially above their respective diagonals.

Finally, both scatter diagrams in Figure 5 strongly suggest a strong, positive, monotonic relation between SMCS and PMCS. The strength of this relation can be quantified by Kendall’s \( \tau_b \) statistic,
Figure 5: Scatter diagrams of how successfully SMCS and PMCS performed on 100 realizations of $f_2$ and 100 realizations of $f_1$. Each point corresponds to a single $f_p$. The horizontal and vertical coordinates of a point are the numbers of 1000 runs of SMCS and PMCS that were successful. A run was considered successful if a function value less than the second smallest local minimum of $f_p$ was returned.

or, equivalently, by the more easily interpreted proportion of concordant pairs statistic. Draw a pair of objective functions from either bivariate sample. A pair is concordant if the same algorithm was more successful for each function in the pair. If there was no monotonic relation between SMCS and PMCS, then the theoretical probability of drawing a concordant pair would be .5. In fact, the observed proportion of concordant pairs is .700 for $p = 2$ and .889 for $p = 4$. A standard statistical test reveals that one can decisively reject the null hypothesis of no monotonic relation. This conclusion is important because it suggests that there is an objective basis for stating that some of the $f_p$ are more difficult to globally optimize than are others.

We conclude with a warning against overinterpretation. Although our results reveal that PMCS is dramatically better than SMCS for the two populations studied, one should not leap to the conclusion than PMCS will be better than SMCS for other populations. In fact, we performed several informal experiments that suggested that SMCS will be better than PMCS for $f_5$ drawn with the same parameter settings. This is not terribly surprising, as the number of function
evaluations required for pattern search to converge increases with the dimension of the problem. For \( p = 5 \), it appears that SMCS typically completes one compass search and barely begins a second before exhausting its budget of \( V = 1000 \) function evaluations. The budgets of \( V_i = 99 \) function evaluations that control the 10 compass searches begun by PMCS are so stringent that these searches typically terminate before substantial progress has been made. Conversely, one can imagine circumstances in which compass search typically converges before exhausting a budget of 99 function evaluations. In such circumstances, PMCS would not exhaust its total budget, SMCS would conduct more than 10 compass searches, and we would again expect SMCS to outperform PMCS.

It is not difficult to invent variants of SMCS and/or PMCS that might perform better than the two implementations that we have studied. We might consider budgets other than \( V = 1000 \) and/or more or fewer than 10 compass searches. We might also adopt a less stringent criterion for convergence. However, any such variant would be an algorithm other than the two that we have studied. The reality of computational experimentation is that one can only draw formal inferences about the algorithms that one has specified and the populations that one has sampled. Experiments far more comprehensive than the ones that we have performed would be needed to comment definitively on the general pros and cons of serial multistart versus parallel multistart. We only hope that we have indicated some useful methods whereby such experiments (and many others) can be undertaken.

Acknowledgments

Michael Trosset was supported by a gift from the Upstream Strategic Research Center of the Mobil Technology Company; and by the National Aeronautics and Space Administration, under NASA Contract No. NAS1-97046, while in residence at the Institute for Computer Applications in Science and Engineering (ICASE), NASA Langley Research Center. Anthony Padula was supported by Virginia Torczon’s Career Development Award from the National Science Foundation. We thank Elizabeth Dolan, Adam Gunson, Erin Parker, Anne Shepherd, Christopher Siefert, and Virginia Torczon for their use and constructive criticism of the methods described herein.

References


