A Rigorous Framework for Optimization of Expensive Functions by Surrogates

Andrew J. Booker [*] J. E. Dennis, Jr. [†] Paul D. Frank [‡]
David B. Serafini [§] Virginia Torczon [¶] Michael W. Trosset [‖]

November 20, 1998

Abstract

The goal of the research reported here is to develop rigorous optimization algorithms to apply to some engineering design problems for which direct application of traditional optimization approaches is not practical. This paper presents and analyzes a framework for generating a sequence of approximations to the objective function and managing the use of these approximations as surrogates for optimization. The result is to obtain convergence to a minimizer of an expensive objective function subject to simple constraints. The approach is widely applicable because it does not require, or even explicitly approximate, derivatives of the objective. Numerical results are presented for a 31-variable helicopter rotor blade design example and for a standard optimization test example.

Key Words: Approximation concepts, surrogate optimization, response surfaces, pattern search methods, derivative-free optimization, design and analysis of computer experiments (DACE), computational engineering.

[*]Mathematics & Engineering Analysis, Boeing Shared Services Group, Applied Research and Technology, Box 3707, M/S 7L-22, Seattle, WA 98124.

[†]Department of Computational and Applied Mathematics & Center for Research on Parallel Computation, Rice University, P. O. Box 1892, Houston, TX 77005.

[‡]Mathematics & Engineering Analysis, Boeing Shared Services Group, Applied Research and Technology, Box 3707, M/S 7L-21, Seattle, WA 98124.

[§]National Energy Research Scientific Computing Center, E.O. Lawrence Berkeley National Laboratory, MS 50B–2239, 1 Cyclotron Road, Berkeley, CA 94720, <dbs@nersc.gov>.

[¶]Department of Computer Science, College of William & Mary, P. O. Box 8795, Williamsburg, VA 23187.

[‖]Department of Mathematics, College of William & Mary, P. O. Box 8795, Williamsburg, VA 23187.

1 Introduction

The use of computer simulations in engineering decision-making is growing in importance. A prototypical example, described in Section 3, involves designing a low-vibration helicopter rotor blade. This example poses an optimization problem in which evaluation of the objective function requires running expensive analysis code(s). Existing methods for such optimization problems are either impractical or *ad hoc*. In this paper, we present a rigorous framework for optimizing expensive computer simulations through the use of inexpensive approximations of expensive analysis codes.

We will set forth, for comment and criticism, a rigorous approach to solving the following mathematical problem:

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & x \in \mathcal{B} \equiv \{x \mid a \leq x \leq b\}, \end{array} \tag{1}$$

where $f : \Re^n \rightarrow \Re \cup \{\infty\}$, $a, b \in \Re^n$, and $a \leq b$ means that each coordinate satisfies $a_i \leq b_i$. The following characteristics distinguish the subset of such problems for which our methods are intended:

1. The computation of $f(x)$ is very expensive and the values obtained may have few correct digits.

2. Even if x is feasible, the routines that evaluate $f(x)$ may fail to return a value at the same computational cost as if a value were returned.

3. It is impractical to accurately approximate derivatives of f.

4. If x is infeasible, then $f(x)$ may not be available.

Typically, $f(x)$ is expensive to evaluate because there are large numbers of ancillary or system variables that must be determined for each choice of x before $f(x)$ can be evaluated. For example, in the helicopter rotor blade design problem, each x specifies a coupled system of partial differential equations (PDEs) that must be solved in order to obtain dependent system variables required to evaluate $f(x)$. It may be quite difficult to obtain accurate solutions of such systems, even after expending substantial computational resources. Furthermore, when a coupled system of PDEs is solved by an iterative method, e.g. the notoriously unreliable method of successive substitution, the method may fail to converge at all. Thus, one cannot assume even that one will obtain an objective function value at each feasible point.

The difficulties implied by the first two properties are compounded if the intended optimization algorithm requires derivative information. Actual derivatives are rarely available, although we hope that this circumstance will change as automatic differentiation technology advances. On the other hand, choosing an appropriate step size for approximating derivatives by finite differences is itself a difficult undertaking. Moreover, the difficulties are compounded by the expense of function evaluation and the fact that a function value may not be returned.

The foregoing considerations lead us to dismiss the possibility of using traditional quasi-Newton methods to solve Problem (1). Indeed, even if actual derivative information were available, quasi-Newton methods might be poor choices because they are adversely affected by function inaccuracies [15]. Instead, we observe that direct search methods [13, 38, 42, 40] do not require derivatives and are relatively insensitive to function inaccuracies. Their shortcoming, especially when function evaluation is expensive, is that in practice they tend to require a great many function values. The essential observation of the present paper is that inexpensive surrogate objective functions can be used to accelerate (certain methods of) direct search for a solution without sacrificing theoretical guarantees of asymptotic convergence.

The use of direct search methods provides a natural way to address the fact that $f(x)$ may be unavailable for some feasible x. When this occurs, we simply assign $f(x) = \infty$. This assignment implicitly assumes that x is suboptimal if $f(x)$ is not available. In fact, the failure of $f(x)$ to evaluate might result from failures in the analysis code rather than from the physical suboptimality of the design x, but we do not attempt to distinguish these possibilities in the present paper.

If the optimization method causes us to consider an infeasible x, then we decline to try to evaluate $f(x)$. In fact, it is common practice for optimization algorithms not to evaluate the objective at points that violate simple bound constraints because such violations are easily detected. In contrast, it is also common practice to evaluate the objective whenever the bound constraints are satisfied, regardless of the possible violation of more complicated (e.g. nonlinear equality) constraints. In the present paper, we skirt this issue by considering formally only bound constraints. Of course, we are keenly aware that most problems also include other types of constraints, but the rigorous management of such constraints is a topic for future research—to consider it now would only cloud the issues that we wish to address in this forum. Thus, we treat the linear inequality constraints in our helicopter rotor blade test example by declining to evaluate $f(x)$ when x is infeasible.

Problems of the type that we have described arise in disparate ways in engineering design and in manufacturing process control. Furthermore, there is a standard engineering practice [1] for attacking such problems:

1. Choose a surrogate s for f that is either
   (a) a simplified physical model of f; or
   (b) an approximation of f obtained by evaluating f at selected design sites, $x_1, \ldots, x_d \in \mathcal{B}$, at which each $f(x_i)$ is finite, then interpolating or smoothing the function values thus obtained.
2. Minimize the surrogate s on $\mathcal{B}$ to obtain x_s.

3. Compute $f(x_s)$ to determine if improvement has been made over the best x found to date, which may be some baseline x or one of the design sites (if that approach is used).

The standard practice violates a fundamental tenet of numerical optimization, that one should not work too hard until one nears a solution. In fact, the standard practice is a one-shot approach: except for the final validation of x_s, all of the function evaluations are performed at sites selected by experimental design criteria with no concern for optimization *per se*. Furthermore, this approach begs a potentially embarrassing question, viz., what does one do if (as is often the case) x_s is not good enough to use as a solution to Problem (1)?

A natural modification of the standard practice is to use a sequence of surrogates to identify promising regions in which to use successively better surrogates, either by adopting models with greater physical fidelity or by constructing approximations from a greater concentration of design sites. Examples of this basic strategy include [6, 20, 14].

We present here a general methodology inspired by ideas in [14]. Our methodology is built on top of a general class of direct search methods for numerical optimization, the *pattern search* methods. We exploit in a novel way the convergence analysis for pattern search methods presented in [39, 24, 25]. Key to our approach is the observation that the convergence analysis allows great flexibility in the heuristics that one can employ to find the next iterate. Accordingly, we perform a fairly extensive search on the current surrogate to select new points at which to evaluate the objective. In this paper, we are concerned with surrogates that are interpolating approximations of the objective and we use any new values that we obtain to update the current approximation.

In the next section, we present our surrogate management framework (SMF) and demonstrate that it works on a standard test problem from the global optimization literature [16]. In subsequent sections, we elaborate on the earlier presentation by examining a problem for which the computational cost of evaluating the objective can be substantial. In Section 3, we describe the helicopter rotor blade design problem. In Section 4, we describe a family of interpolating approximations that has become popular in the literature on the design and analysis of computer experiments (DACE). In Section 5, we sketch some ways of using DACE approximations as optimization surrogates. Finally, in Section 6, we report some numerical results.

Some indications of how this work fits into a larger effort are provided in Sections 2.3 and 7.

2 A Rigorous Framework for Optimization Using Surrogates

In this section we describe SMF, our framework for managing surrogate objective functions to facilitate the optimization of expensive computer simulations. The framework is sufficiently general to accommodate surrogates that are (1) simplified physical models of the expensive simulation; (2) approximations of the expensive simulation, constructed by interpolating or smoothing known values of the objective; or (3) model-approximation hybrids. For the sake of clarity, however, we focus on the case of surrogates of the second type. In Section 4 we will emphasize interpolating approximations constructed by kriging, but the simple example in Section 2.3 illustrates that SMF also works with polynomial interpolants. Another example using polynomial interpolants is presented in [32].

We begin, in Section 2.1, by describing the family of underlying optimization algorithms on which SMF is based. Next, in Section 2.2, we formally define SMF. We conclude, in Section 2.3, by using SMF to minimize a simple algebraic test function.

2.1 Pattern Search Algorithms

Pattern search algorithms are a class of direct search methods for numerical optimization. A formal definition of pattern search, which includes various well-known algorithms, was proposed in [39]. An elementary introduction to pattern search algorithms and a discussion of their historical antecedents is available in [40].

Pattern search algorithms are characterized by two crucial notions, a sequence of *meshes* and a list of *polling conditions*. A mesh is a lattice to which the search for an iterate is restricted. As optimization progresses, the polling conditions govern when the current mesh can be refined, ensuring that the algorithm will satisfy the demands of the convergence theory for pattern search methods.

For our purposes, the primary polling condition that must be enforced to ensure convergence is that the set of vectors formed by taking the differences between the set of trial points at which the objective function is to be evaluated (the pattern) and the current iterate x_k must contain a positive basis for $\Re^n$. A positive basis [11] is a set of vectors whose nonnegative linear combinations span $\Re^n$, but for which no proper subset has that property. For our purposes, the relevance of a positive basis is that it ensures that if the gradient of f at x_k is not zero, then at least one vector in the positive basis defines a descent direction for f from x_k. This can be guaranteed without any knowledge of the gradient. Any positive basis has at least $n+1$ and at most $2n$ vectors; we call these minimal and maximal positive bases, respectively.

For unconstrained problems, a minimal positive basis is sufficient to guarantee convergence [25]. However, for problems with rectangular feasible regions, e.g. Problem (1), we use a maximal positive basis that comprises all of the coordinate directions, both positive and negative. This guarantees that it is possible to move along the boundary of the feasible region and thus prevents premature convergence to a point that is not a constrained stationary point [24]. Recent work [26] has revealed that it is possible to construct adaptive pattern search algorithms that identify only those constraints that are either binding or "almost" binding at the current iterate so that the number of vectors needed at any given iteration can vary between $n+1$ and $2n$, inclusively.

The following formulation of Generalized Pattern Search (GPS) differs from the formulation of pattern search in [39, 24, 25, 26], but it is especially well-suited to our presentation.

We remind the reader that if $f(x)$ either is infeasible or cannot be evaluated successfully, then we set $f(x) = \infty$.

GPS: Let M_0 denote a mesh on $\mathcal{B} \equiv \{x \mid a \leq x \leq b\}$ and suppose that $x_0 \in M_0$ has been given. (In typical practice, $x_0 \approx x^*$, where x^* is a preliminary baseline solution, but any choice of $x_0 \in M_0$ is possible.) Let $X_0 \subset M_0$ contain x_0 and any $2n$ points adjacent to x_0 for which the differences between those points and x_0 form a maximal positive basis (composed of multiples of the coordinate vectors) for $\Re^n$. As the algorithm generates $x_k \in M_k$, let $X_k \subset M_k$ be defined in the same way. For $k = 0, 1, \ldots$, do:

1. **Search**: Employ some finite strategy to try to choose $x_{k+1} \in M_k$ such that $f(x_{k+1}) < f(x_k)$. If such an x_{k+1} is found, declare the **Search** successful, set $M_{k+1} = M_k$, and increment k;

2. else **Poll**:
   if x_k minimizes $f(x)$ for $x \in X_k$, then declare the **Poll** unsuccessful, set $x_{k+1} = x_k$, and refine M_k to obtain M_{k+1} by halving the mesh size (write this as $M_{k+1} = M_k/2$); else declare the **Poll** successful, set x_{k+1} to a point in X_k at which $f(x_{k+1}) < f(x_k)$, and set $M_{k+1} = M_k$.
   Increment k.

Step 2 provides the safeguard that guarantees convergence, as in the following result [24].

Theorem 2.1.1. If f is continuously differentiable on the feasible region $\mathcal{B}$, then some limit point of the sequence $\{x_k\}$ produced by a generalized pattern search (GPS) method for bound constrained minimization is a constrained stationary point for problem (1).

Notice that this result guarantees that GPS will converge no matter how naive the search strategy in Step 1. In practice, of course, the sophistication of the search strategy matters a great deal. We now turn to SMF, which uses surrogate objective functions to try to **Search** with greater parsimony and thereby reduce the total number of objective function evaluations.

2.2 The Surrogate Management Framework

The description of SMF that we present here is a set of strategies for using approximations in both the **Search** and **Poll** steps of a GPS algorithm. For greater clarity, we have also identified a separate **Evaluate/Calibrate** step. In what follows, we assume that a family of approximating functions has been specified, that an initial approximation has been constructed, and that an algorithm to recalibrate the approximation is available.

SMF: Given s_0, an initial approximation of f on $\mathcal{B}$, and $x_0 \in M_0$, let $X_0 \subset M_0$ contain x_0 and any $2n$ points adjacent to x_0 for which the differences between those points and x_0 form a maximal positive basis (composed of multiples of the coordinate vectors) for $\Re^n$. As the algorithm generates $x_k \in M_k$, let $X_k \subset M_k$ be defined in the same way. For $k = 0, 1, \ldots$, do:

1. **Search**: Use any method to choose a trial set $T_k \subset M_k$. If $T_k \neq \emptyset$ is chosen, then it is required to contain at least one point at which $f(x)$ is not known. If $T_k = \emptyset$, then go to **Poll**.

2. **Evaluate/Calibrate**: Evaluate f on elements in T_k until either it is found that x_k minimizes f on T_k or until $x_{k+1} \in T_k$ is identified for which $f(x_{k+1}) < f(x_k)$. If such an x_{k+1} is found, then declare the **Search** successful. Recalibrate s_k with the new values of f computed at points in T_k.

3. If **Search** was successful, then set $s_{k+1} = s_k$, $M_{k+1} = M_k$, and increment k;
   else return to **Search** with the recalibrated s_k, but without incrementing k.

4. **Poll**:
   If x_k minimizes $f(x)$ for $x \in X_k$, then declare the **Poll** unsuccessful, set $x_{k+1} = x_k$, and set $M_{k+1} = M_k/2$;
   else declare the **Poll** successful, set x_{k+1} to a point in X_k at which $f(x_{k+1}) < f(x_k)$, and set $M_{k+1} = M_k$.
   Recalibrate s_k with the new values of f computed at points in X_k. Set $s_{k+1} = s_k$.
   Increment k.

We structure our discussion of SMF around the proof of the following corollary of Theorem 2.1.1. Notice that this result assumes nothing about the accuracy of the approximations. In practice, of course, we would expect better approximations to yield better results.

Theorem 2.2.1. If f is continuously differentiable on the feasible region $\mathcal{B}$, then some limit point of the sequence $\{x_k\}$ produced by SMF for bound-constrained minimization is a constrained stationary point for problem (1).

Proof: The proof is accomplished by showing that SMF is an instance of a generalized pattern search method and so Theorem 2.1.1 applies.

First, we need to be sure that we have specified a finite **Search** step, i.e., that there is a fixed upper bound on the number of unsuccessful search steps that will be tried before a poll step is taken. This follows immediately because each choice of $T_k \neq \emptyset$ must contain at least one point of M_k at which f is unknown and M_k is a mesh on a compact set $\mathcal{B}$, hence a finite set.

We finish the proof by noting that the **Poll** step is still intact. From the perspective of the optimization algorithm, **Poll** is functionally unchanged by the recalibration step that it now includes. $\square$

The key to a successful implementation of SMF is to define the **Search** strategy in a way that efficiently exploits the current approximation s_k. One obvious approach is to search for points that minimize s_k. In [41], for example, a finite-difference quasi-Newton method was started from the current iterate with s_k as the objective function. A more ambitious strategy would be to explore s_k globally for multiple prospective basins, e.g. by performing a comprehensive grid search. For the examples in this paper, we performed a comprehensive search on a subset of the current mesh, M_k.

Notice that we do not require f to be evaluated at all points in T_k or X_k before declaring a successful **Search** or **Poll** step. Once we have identified a point in either T_k or X_k with an objective value strictly less than $f(x_k)$, we can declare the iteration to be successful and increment k. This practical flexibility derives from a powerful and crucial aspect of the convergence theory developed in [39, 24, 25, 26]: it is not necessary for a pattern search algorithm to find the best point on the current mesh M_k, or even the best point in X_k—any point that produces decrease on $f(x_k)$ will suffice.

The convergence theory states that any point on the current mesh, M_k, that produces decrease on $f(x_k)$ can be used as the next iterate, x_{k+1}. Traditional pattern search algorithms evaluate f at a predetermined subset of M_k (a *pattern*) in order to try to discover such points. To try to reduce the number of function evaluations required to discover a point that produces decrease, SMF uses the current approximation to predict points in M_k at which we expect to realize decrease. The set T_k contains our list of potential candidates. If T_k contains multiple candidates, then we choose those that are considered most promising. If the approximation does not predict any such decrease, then we may choose to set $T_k = \emptyset$ and **Poll**.

SMF affords complete flexibility in deciding how many points to include in T_k. One obvious possibility is to include a single point: the one at which the surrogate predicts the greatest decrease on $f(x_k)$. This is precisely what we did to obtain the results reported in Section 2.3. In other situations, however, it may be desirable to include several points. One such circumstance arises when attempts to evaluate the objective are prone to failure. For example, for the helicopter rotor blade design problem we have performed runs in which 60% of our attempts to evaluate f at a feasible x failed. Thus, to obtain the results reported in Section 6, we choose T_k to contain three points in an effort to ensure that at least one of the points in T_k can be evaluated successfully. Another circumstance arises in parallel or distributed computing environments. If several processors are available to perform simultaneous function evaluations, then it is natural to provide T_k with one point for each available processor.

We also allow T_k to contain points at which the approximation does not predict decrease. This flexibility is desirable because the step **Evaluate/Calibrate** actually serves two purposes. On the one hand, we obviously want to find an $x_{k+1} \in T_k$ for which $f(x_{k+1}) < f(x_k)$. On the other hand, after each step we know more objective function values and we therefore compute a new (and presumably more accurate) approximation. However, the points that most decrease the objective may not be the points that most improve the accuracy of the approximation. In fact, it may be desirable to select trial points that balance the competing goals of decreasing the objective and constructing a better approximation. By selecting trial points that lead to better approximations, we may gain greater insight into the global behavior of f, accelerating optimization by improving the quality of future searches. This idea will be discussed in Section 5.

Except for recalibration, the **Poll** steps for GPS and SMF are identical. Although SMF does not specify the order in which f is to be evaluated at the points in X_k, it is natural to evaluate them in increasing order of the objective values predicted by s_k.

If the current iterate is sufficiently near a minimizer of the true objective function, then the current mesh must be refined in order for optimization to progress. It is the **Poll** step that guarantees convergence, but this guarantee can be costly. For bound-constrained optimization, an unsuccessful **Poll** step requires between n and $2n$ evaluations of the objective function [25], so we would like to avoid **Poll** steps whenever possible. One possibility is to employ a hybrid approach that starts with SMF and assumes that an unsuccessful **Poll** step signals a basin of attraction for a local method, which we then call to see if it can succeed. Of course, we might need to revert to SMF if we switch too hastily. A natural candidate for the second phase of such a hybrid approach is the derivative-free optimization (DFO) algorithm described in [8, 7]. The development of an SMF-DFO hybrid is one objective of the larger effort mentioned in Sections 2.3 and 7.

2.3 Sample Test Results

We now apply SMF to a standard global optimization test problem, the six-variable Hartman problem [16], which has a single global minimizer and several nonglobal minimizers. In realistic applications, objective functions are expensive to evaluate and computed values have only several digits of accuracy. Hence, we only attempt to solve the Hartman problem approximately, and we are prepared to decrease the chance of converging to the global minimizer in order to restrict the total number of objective function evaluations.

Figure 1 presents run histories for two implementations of SMF, as well as the final values obtained from nine runs of the DFO algorithm described in [8, 7]. We imposed bounds of $\mathcal{B} = [0, 1]^6$ and started each run from $x_0 = (.5, .5, .5, .5, .5, .5)^T$. (The nine runs of DFO produced different results because the DFO algorithm includes a stochastic decision.) The only difference between the two implementations of SMF is the choice of approximating families: one choice interpolated known function values with variable-order multivariate polynomials [12] whose degrees were increased as more function values were obtained; the other choice interpolated known function values by kriging. The latter family of approximations, which we also used for the helicopter rotor blade design problem, is discussed in Section 4. In each implementation, the initial approximation was constructed by interpolating the same set of 16 known function values. Each time that **Search** was called, the current approximation was evaluated on a 7280-point subset of the current mesh. Because the Hartman objective can always be evaluated, **Search** returned only a single point from the subgrid at which the approximation predicted the greatest decrease in the Hartman objective.

The global minimum of the Hartman function is -3.322. Except for one run of DFO, each run produced (approximately) this value. Of particular note is the fact that SMF worked effectively with each of two different families of approximations. The SMF run histories exhibit the characteristic plateaus that result when **Poll** steps are executed on a sequential computer. The actual ("wall clock") time spent on these steps can be reduced if opportunities for parallel or distributed computing exist. If a sufficient number of processors are available to evaluate all of the required function values concurrently, and if the abscissa indicates actual time rather than number of function evaluations, then the plateaus usually disappear.

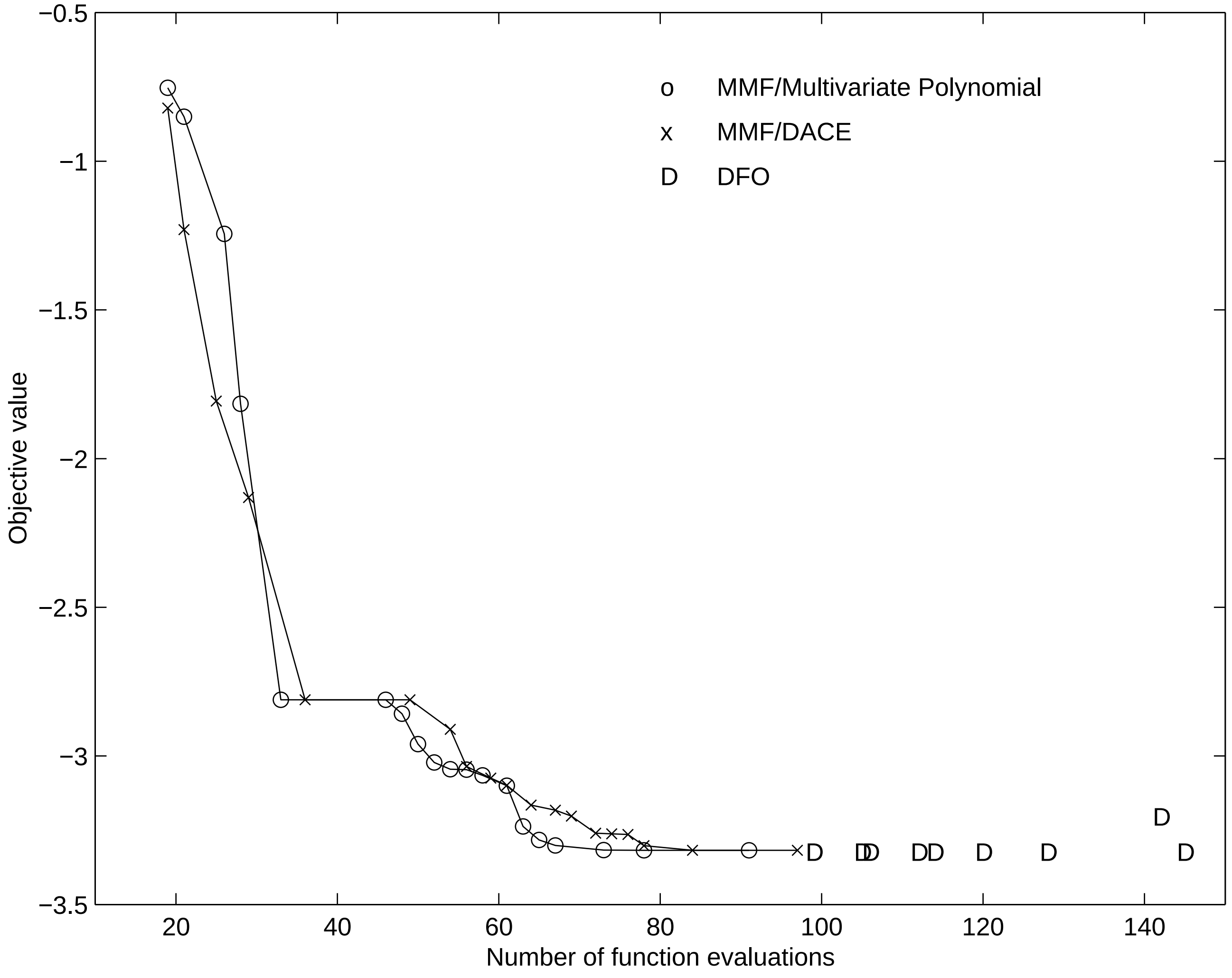

Figure 1: Results for the six variable Hartman problem

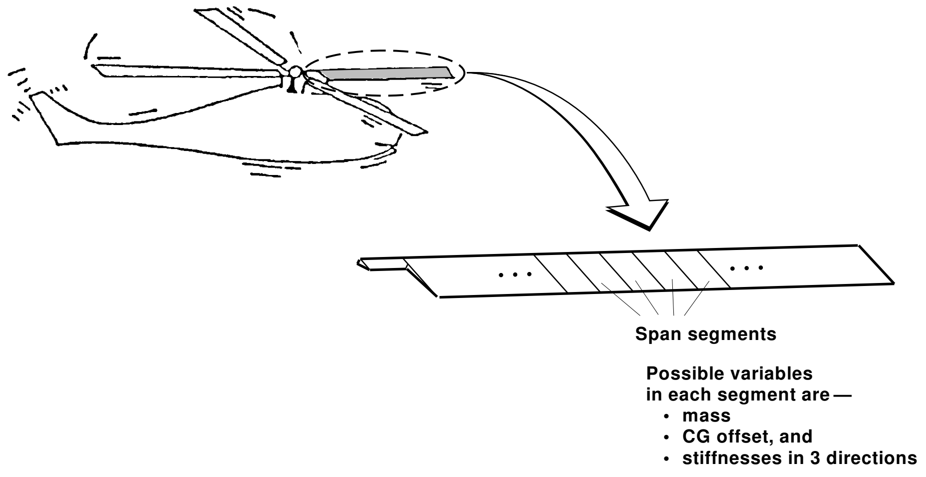
Span segments
Possible variables in each segment are —
• mass
• CG offset, and
• stiffnesses in 3 directions

Figure 2: Rotor blade design variables

3 The Helicopter Rotor Blade Design Problem

Helicopter rotor blade design is used here for illustrating concepts and motivating algorithmic requirements for SMF applied to industrial problems. The particular task we consider is structural design of helicopter rotor blades for minimum vibration transmitted to the hub.

As indicated in Figure 2, the design variables consist of up to five structural parameters for each span segment. The variations on this problem that we have considered have between 10 variables and 56 variables. As described below the objective function is a weighted sum of various harmonics of forces and moments. The analysis code used is Tech01 [33].

Tech01 is a multidisciplinary analysis code. The disciplines include dynamic structures, aerodynamics, wake modeling, and controls. The run time for a Tech01 fixed-wake analysis is roughly 20 minutes on a mid-level workstation. However, the run time can increase to several days on the same machine if wake updating is invoked. The full wake analysis has greater fidelity to the physics of the problem. Our main focus is on the use of approximations to the analysis code results as objective function surrogates for optimization. Thus, to facilitate studies of algorithmic issues for surrogate optimization, the test results discussed here use fixed-wake analyses.

A more detailed statement of the optimization problem is

$$\begin{array}{ll} \text{minimize} & f(x) = \sum_{i=1}^{nh} w_i \dfrac{|h_i(x)|}{|h_i(x_B)+1|} \\ \text{with respect to} & x \in \Re^n \\ \text{subject to} & \mathrm{xu}_j \geq x_j \geq \mathrm{xl}_j,\ j = 1, \ldots, n \\ & \mathrm{cu}_k \geq c_k(x) \geq \mathrm{cl}_k,\ k = 1, \ldots, ncon. \end{array} \tag{2}$$

In the above equation, the $h_i,\ i = 1, \ldots, nh$, are output responses from Tech01. The subscript i is an index that maps from the response function vector elements to forces in three

Example	Variables	Objective	Constraints
31 Variable	10 masses, 10 centers of gravity, 11 stiffnesses in a single direction	weighted sum of 1st and 2nd harmonics for two flight conditions	upper bound on sum of masses

Table 1: Rotor blade design example

directions, moments in three directions, and harmonic numbers for each force and moment. In addition, the indexing may span several flight conditions, such as hover and forward flight at various speeds. Normalization to account for the different physical units of the responses is accomplished by including $h_i(x_B)$ in the denominator of the objective function terms, where x_B is the *baseline* design.

The objective function components are weighted by factors w_i. The limits xu_j, xl_j, cu_k, and cl_k are upper and lower bounds on the variables and constraints, respectively. The constraints $c_k(x)$ can be quantities such as required rotor horsepower, centrifugal force, autorotational inertia, snow load, and limits on total mass. Aside from the bounds on the independent variables, the only constraint in the examples considered here is total mass. Since the masses are a subset of the design variables, the mass constraint is a linear constraint involving a subset of the variables. Thus, it is independent of the analysis results, and does not require consideration of issues involving the construction of surrogate approximations of constraint functions.

The helicopter rotor blade design problem is summarized in Table 1. Note that this problem has upper and lower bounds on all the variables.

4 Constructing the Surrogates

Even with fixed wake, the helicopter rotor blade design examples are not easily modeled using simplified, less expensive simulations. In consequence, we concentrate on approximations constructed by interpolating or smoothing a set of known objective function values. In this section we describe our method of choosing a set of initial design sites at which the objective function f is evaluated before optimization commences, our choice of a class of function approximations from which the initial surrogate s is to be selected, our method of selecting s from this class, and some diagnostic procedures for extracting useful information from s.

The problem of choosing a set of initial design sites, $x_1, \ldots, x_d \in \mathcal{B}$, is a problem in the design of experiments. This problem has been studied extensively in the recent literature on the design and analysis of computer experiments (DACE), surveys of which include [31, 2, 22].

We seek designs that are "space-filling" (for lack of a better term), i.e. that will allow us to sample the behavior of the objective function throughout the feasible region. We want to avoid designs that are tied to a narrow class of approximating functions, e.g. linear or quadratic functions. We want to be able to generate designs somewhat automatically, and we would like to be able to generate designs for irregular (nonrectangular) feasible regions.

We have opted for designs that are used in quasi-Monte Carlo integration: Latin hypercube sampling (LHS) [27, 35], orthogonal arrays ($\mathcal{OA}$) [28] and $\mathcal{OA}$-based LHS [36]. In LHS, each of the n variables is chosen from d equally spaced values. The $\mathcal{OA}$s we use are space-filling in the following sense: the variables in the experimental design are assigned from l distinct values. In every subset of k variables every one of the l^k combinations of values occurs the same number of times. Arrays with this property are of *strength* k. LHS designs are of strength 1. Typically we use $\mathcal{OA}$s of strength 2. This is a straightforward process because LHS designs are easily generated and efficient code for generating $\mathcal{OA}$ designs is available from STATLIB (`http://lib.stat.cmu.edu`).

LHS and $\mathcal{OA}$ were devised for rectangular regions. In the helicopter rotor blade design examples, the mass constraint induces a nonrectangular feasible region. We have experimented with various strategies for adapting $\mathcal{OA}$ designs to this region, e.g.

- Generate a design with d points in the rectangle defined by the variable bounds, then alter the design so that the d points satisfy the mass constraint.

- Generate a design with many points in the rectangle defined by the variable bounds, then discard the points that are outside a slightly expanded mass constraint boundary.

After the design sites have been chosen and the objective function f has been evaluated at them, the initial surrogate s can be constructed. This surrogate is intended to be an approximation of f throughout the region of interest that is inexpensive to evaluate. It will be recalibrated as new function values are obtained in the course of solving the optimization problem. Because we do not want to make *a priori* assumptions about the structure of f, we require a large, flexible class of functions from which surrogates can be selected.

Plausible families of approximating functions include neural networks and low degree interpolating polynomials [12]. In §2.3 we gave evidence that the SMF can use different families of approximation. We have opted for the family of functions defined by the kriging procedures discussed in the DACE literature. The kriging parameterization, defined by means and covariances of function values, is more intuitive for the present applications than other approximations in the response surface literature. For some choices of covariance function, kriging is equivalent to spline interpolation, a correspondence that has been discussed in the geostatistics literature [43].

It is quite common in the statistics literature to motivate kriging by assuming that f is a realization of a stationary Gaussian spatial process. As implausible as this assumption may seem in the present context, it does suggest useful ways to proceed with the selection of a surrogate objective function from the family of approximating functions. Upon making this assumption, it becomes possible to estimate mean and covariance parameters from $f(x_1), \ldots, f(x_d)$ by the method of maximum likelihood estimation (MLE) and thereby to specify a well-defined procedure for selecting s.

Although MLE has been criticized in the spatial statistics literature, e.g. [30], it has been defended by others as a crude form of cross-validation [19, 10]. Our experience to date has been similar to that reported in [31]: “crude MLE’s lead to useful prediction. . . .” Assuming that the covariances in question are a constant unknown variance times unknown correlations of a specified form, there exist closed-form expressions for the MLEs of the mean and variance parameters. To obtain MLEs of the correlation parameters, we have attempted global optimization of the (log) likelihood function via an implementation of the algorithm in [29].

One technical difficulty with kriging should be noted. Kriging calculations require inversion of the matrix of estimated correlations between function values at the design sites. The initial correlation matrix usually is well-conditioned but as the function is sampled at additional sites that cluster near a minimizer, the process of recalibration generally causes subsequent correlation matrices to become ill-conditioned. We have addressed this difficulty by adding a small number (10^{-6}) to the diagonal of the correlation matrix. With this addition, the approximating functions do not exactly interpolate the observed function values; however, they retain their flexibility and predict observations very closely.

Once a surrogate function s has been constructed, one can use it to predict values of $f(x)$ and also to approximately bound the errors in such predictions. The latter is accomplished by calculating mean squared error (MSE) under the assumption of a stationary Gaussian process. It has been argued in [31, 21] that this is a reasonable framework in which to bound future prediction errors, particularly if one can assess the plausibility of the assumption of a stationary Gaussian process. Since larger values of MSE are associated with larger uncertainty in prediction, we have used MSE to guide our choice of new sites at which data would be of particular value in improving the accuracy of the surrogate.

One also might predict future prediction errors by examining the *cross-validation residuals*. These error estimates are obtained at each observation by kriging (with the original MLE parameters) the other observations and predicting the designated observation. Similarly, it was suggested in [21] that one might cross-validate the MSEs to assess their predictive capabilities.

Finally, we have found that performing a functional analysis of variance [17, 28, 31] on the surrogate function s is a useful way of identifying lower-dimensional subspaces in which most of the variation in s resides. This ANOVA technique, which can in principle be applied to any square-integrable function, decomposes s into *main effects* (contributions of individual variables to variation in s) and *interaction effects* (contributions of combinations of variables to variation in s). The hope is that one can identify a few key variables that account for most of the variation in f, then optimize solely with respect to those variables at reduced expense.

5 DACE Model Refinement by Balanced Searches

The **Search** phase of SMF allows us to use *any* method to choose a trial set of new mesh points at which to evaluate the true objective function. In this section we discuss several search strategies with which we have experimented.

Whatever approximations are used as surrogates for the objective function, a straightforward search strategy is to optimize (either locally or globally, depending on one's goals) the current approximation and to return trial points that reside on the mesh M_k near the solution(s) thus obtained.

A simple implementation of this strategy, in which a finite-difference quasi-Newton method was used to find a local minimizer of the current DACE approximation (see §4), is the model-assisted grid search (MAGS) described in [41]. MAGS was intended for situations in which only a relatively small number of function evaluations are permitted. Because it approximates the objective function over the entire feasible region, recalibration of the approximation is made using one new objective value at a time, as these values are produced by the optimization procedure.

In contrast to MAGS, a "zoom-in" method for local refinement was proposed in [18]. This strategy uses the existing approximation to determine an interesting subregion of the design space for further exploration. The optimization process is halted, additional function values are obtained in the subregion, and a new approximation is formed. The expectation is that the approximation constructed in the subregion will be more accurate than the original approximation because it will be based on a higher density of data.

One method for determining the zoom-in region is to locate the local minima of the original approximation and determine the extent of their basins. For a specified value greater than the function value at a local minimum, the extent of its basin is assessed in terms of the distance of the minimum in each coordinate direction to the nearest level set corresponding to the specified value (or the distance to the coordinate bound if the specified level is not attained.)

Zoom-in methods favor exploration in the vicinity of local solutions of the current approximation. Hence, they may fail to find basins of better local solutions elsewhere in the design space. In the parlance of global optimization, they are purely *local* in nature. A purely *global* method for determining new points at which to evaluate f is to minimize an estimate of the integrated mean squared error (IMSE, [31]) of the resulting new approximation. This method is space-filling in that it tends to place new design points in previously unexplored regions. In contrast to zoom-in methods, the IMSE-optimal methods defer examining promising regions in the interest of obtaining a better "global" picture of the design space. The result is that they tend to converge slowly to a minimizer of the objective.

To address both local and global concerns, we have experimented with a *balanced search* strategy. This method is based on our observation that, at any location in design space, the current DACE approximation can supply two key pieces of information: an approximate value of the objective and an estimate of the approximation's mean squared error (MSE) at the point. The former provides purely local information; the latter, which increases with the distance from the subject point to the nearest design site and with the degree of nonlinearity of the data, quantifies uncertainty about the behavior of the true objective function and hence provides some degree of global information.

Based on local concerns, one would evaluate f at points that the approximation indicates have low values. Based on global concerns, one would evaluate f at points with high MSE values. The balanced search method selects a portion of the total of the trial set T_k based on each measure. Ideally, one would measure the approximate values and MSEs at each point on a fine grid in design space and select the best candidates, as in [9]. Unfortunately, in high-dimensional design spaces it is impossible to consider even a crude grid formed by splitting each dimension in two; hence, the balanced search algorithm described below considers each member of a "dense cloud" of (say 5000) trial sites. To ensure that this cloud is space-filling, it is generated from an $\mathcal{OA}$-based LHS.

Balanced Search Algorithm for New Site Selection
Given: an existing approximation, current design sites, a list of local minimizers of the current approximation, the number *nlocal* of new design sites to be based on local concerns, the number *nglobal* of new design sites to be based on global concerns, and a tolerance τ equal to the minimum distance that will be allowed between any two sites.

1. Create an initial list of (say) 5000 trial sites using an $\mathcal{OA}$-based LHS.
2. Add the local minimizers of the current approximation to the list of trial sites.
3. Calculate the distances from each of the trial sites to each other and to each of the design sites for the current approximation.
4. Sort the trial sites in order of increasing values as determined by the current approximation.
5. Select as new design sites the *nlocal* trial sites with the smallest values determined by the current approximation, maintaining the condition that each design site is $\geq \tau$ distant from every other design site.
6. Compute the MSE at each of the remaining trial sites using the correlation parameters of the current approximation, but after updating the model parameters to reflect MSE $= 0$ at the newly selected design sites.
7. For $i = 1, \ldots, nglobal$, do:
    (a) Select as a new design site the trial site with the largest MSE value, maintaining the condition that each design site is $\geq \tau$ distant from every other design site.
    (b) Update the approximation to reflect MSE $= 0$ at the newest design site, then recompute the MSE at each of the remaining trial sites.

6 Test Results for Rotor Blade Design

We now summarize the performance of several optimization methods when applied to the helicopter rotor blade design problem described in Section 3. We remind the reader that this problem has a linear inequality constraint which we treat by declining to evaluate $f(x)$ for any infeasible x. The optimization methods that we considered are the following:

- **MMF**: This is Serafini's [32] implementation of the surrogate management framework, SMF, described in Section 2.2. An initial approximation was constructed from 59 successful function evaluations using the DACEPAC software package [2, 3]. The initial iterate was a baseline solution provided by Boeing. **Search** evaluated the current approximation on a 29,800-point subset of the current mesh and returned the three points with the lowest values. The true objective function was then evaluated sequentially at each of these points until one was found to be better than the current iterate.

- **DFO**: This is the derivative-free optimization method discussed in [7, 8]. The results that we report, which include final values but not run histories, were provided by Katya Scheinberg. The initial iterate was the baseline solution provided by Boeing. Two variants of DFO were implemented, one that scales the decision variables to be of comparable magnitude and one that leaves the decision variables unscaled. (The significance of this distinction will be discussed below.) Because DFO randomly chooses the second point at which the objective function is evaluated, multiple runs of each variant were performed (nine for the scaled variant, ten for the unscaled variant).

- **PDS**: This is Torczon's implementation [37] of the parallel direct search method of [13], with modifications by Serafini to support constraints and the standard Message Passing Interface (MPI) parallel communications library [34]. The initial iterate was the baseline solution provided by Boeing. PDS was executed using 96 evaluations of the objective per iteration, more than the minimal number (62) required to ensure convergence.

- **GA**: This is a genetic algorithm from PGAPack [23]. On the advice of its author, David Levine of the Boeing Company, we used a steady-state reproductive strategy with a population size of 200 and a replacement rate of 10% of the population per iteration.

- **BLGS**: This implementation of the SMF is due to Booker and Frank [4] and was discussed in Section 5. The initial approximation was the same as for MMF. The current approximation was refined twice, each time by adding 50 new values of the objective function. Some of these 50 new sites were chosen because the current approximation predicted that they would have objective function values lower than that of the current iterate; others were chosen because they were relatively far from any previously selected sites.

- **Sampling**: This is a simple sampling algorithm that generates $\mathcal{OA}$-based LHS of the Bose type [28]. Each sample contains the initial design, and in addition, samples that contain 58, 200, 380, and 684 *convergent* points were generated independently. For each sample, the best value of the objective function was taken to be the minimum of the objective function values computed at the points in that sample, if this value improved on the best found in prior samples. We include these results only as a simple strawman, and for this reason we did not count the rather larger number of points in each sample for which the $f(x)$ did not return a value, nor did we try to implement a more sophisticated sampling algorithm.

For each of the above optimization methods, the best objective function value obtained after selected numbers of function evaluations was plotted against the number of function evaluations. The resulting graph, adapted from [5, 32], is displayed in Figure 3. We report the total number of *attempts* to evaluate the objective function, whether or not the attempt was successful. However, we did not count unsuccessful attempts encountered during the construction of initial approximations, prior to commencement of the optimization algorithm. Thus, for MMF and BLGS, our count includes the 59 successful function evaluations obtained by DACEPAC, but not the additional 97 evaluation attempts that failed. For the sampling algorithm, only successful function evaluations were counted.

The results summarized in Figure 3 are quite encouraging—so good, in fact, that it may be that the 31-variable helicopter rotor blade design problem is substantially easier to solve than we anticipated. Both GA and PDS performed as advertised. GA produced substantial decrease with a small number of function evaluations, but then had difficulty descending below a fairly high value of the objective function. PDS descended somewhat more steadily to an appreciably lower value of the objective function. Both DFO and MMF found even lower objective function values in a number of function evaluations that would be considered extremely small for finite-difference quasi-Newton methods.

Except for one variant of DFO, all of the algorithms for which we have reported results scale the decision variables to be of comparable magnitudes. The variables in the 31-variable helicopter rotor blade design problem differ by ten orders of magnitude, yet the single lowest value of the objective function was found by the variant of DFO that did *not* scale the variables. This apparent paradox deserves further comment.

When DFO is applied to the unscaled problem, its trust region precludes appreciable change in the variables of large magnitude but hardly restricts changes in the variables of small magnitude. In effect, DFO thereby restricts its search to the subspace defined by the variables of small magnitude. A subsequent ANOVA decomposition of a DACE approximation of the 31-variable helicopter rotor blade design objective function revealed that the objective does not vary much with respect to the variables of large magnitude. Thus, the unscaled variant of the DFO implementation was actually solving a lower-dimensional problem coincidentally generated by the most important variables. (This is a dramatic illustration of the diagnostic value of the ANOVA decomposition.) We are now investigating the lower-dimensional problem in greater detail.

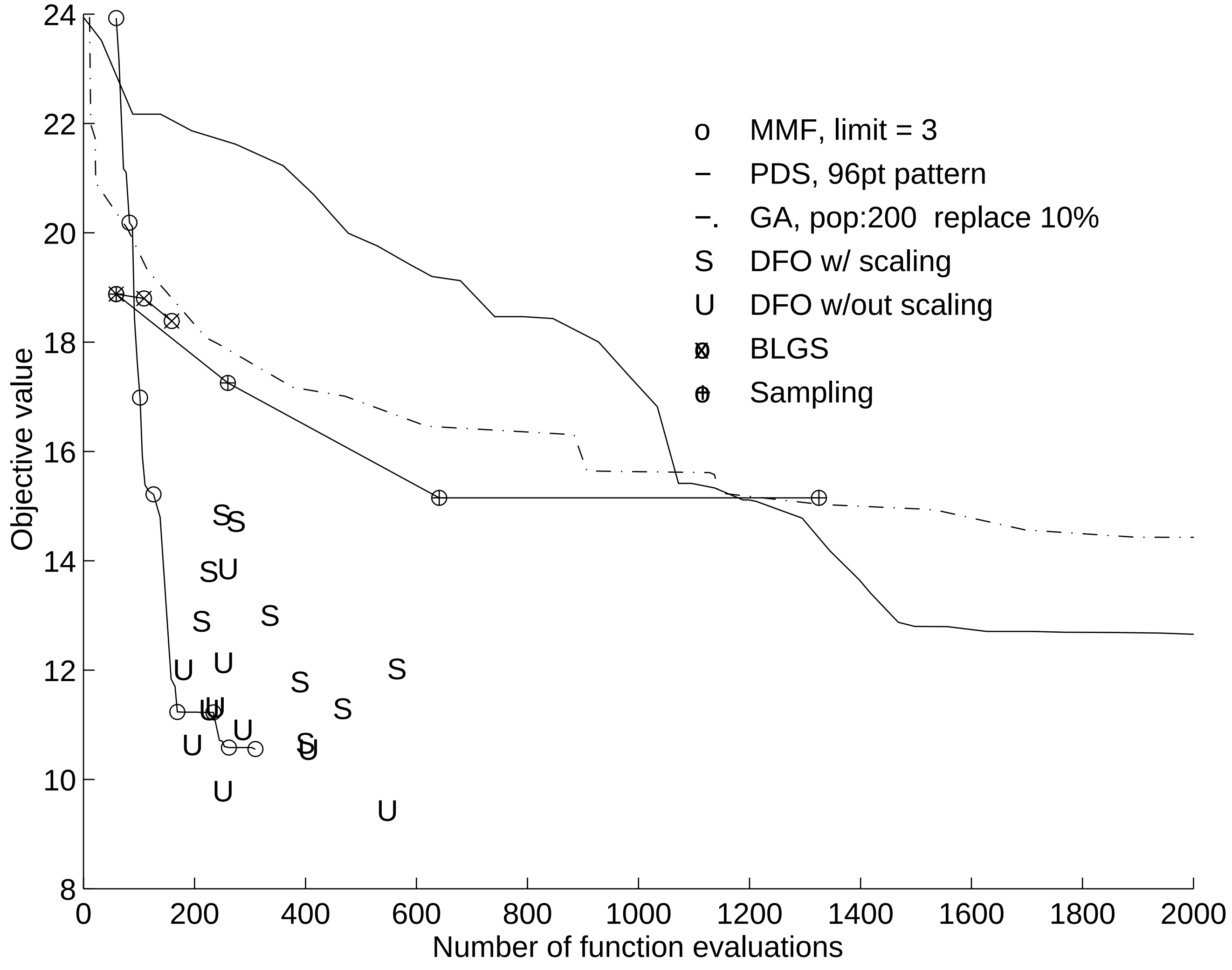

Figure 3: Results for the 31-variable helicopter rotor blade design problem

Finally, we observe that most of the above algorithms can exploit parallelism to reduce the "wall clock" time required to get a solution, principally by concurrent evaluations of the objective function. The implementations used here differ with respect to how many concurrent evaluations can be used effectively. In particular, two of the codes, PDS and GA, were designed explicitly to be executed in parallel and so they have the advantage that they can use any number of processors without any recoding. Given this implementation philosophy, the total number of function evaluations these methods take to reach a solution does not compare favorably with the sequential implementations of the other algorithms. However, when executed in parallel, the "wall clock" time for PDS and GA is more competitive. Nonetheless, for the tests reported here, MMF and some runs of DFO found feasible solutions with appreciably lower values of the objective and required far fewer total evaluations of the objective function in the process.

7 Conclusions

The results reported in Section 6 lend credence to our overall plan to develop approximation-based optimization methods that use SMF. Our current intent is to construct DACE approximations of sufficient accuracy that ANOVA decomposition will provide insight into the problem at hand. Subsequently, some variant of SMF, perhaps one with a BLGS flavor, will be used to identify the basin of a promising minimizer of the true objective function.

Of course, much remains to be done. We would like to find ways to accelerate the search for a minimizer after SMF has identified a basin and its reduction of the objective function has begun to slow. One possibility is then to use known objective function values to provide an initial approximation for DFO, as there are reasons to be believe that DFO enjoys faster local convergence properties than SMF. Another important challenge is to extend SMF to address problems with general constraints, particularly constraints that involve outputs of expensive analysis codes. We are currently working to address these issues.

Acknowledgments

The computational results presented in this paper were enriched significantly by the cooperation and collaboration of Katya Scheinberg in providing DFO test results. Our results showing that SMF can work with polynomial as well as DACE approximations would have been much more difficult without the cooperation of Tom Grandine, who provided his implementation of the least-degree polynomial interpolant [12]. We also thank David Levine for suggesting parameter values for use in PGAPack.

Much of this work was accomplished during the year John Dennis and David Serafini spent in the Mathematics and Engineering Analysis Group at Boeing Shared Services Group Applied Research & Technology Division in Seattle. They wish to thank DOE and AFOSR for support, respectively, under grants DOE FG03–93ER25178 and AFOSR–F49620–95–1–

0210, as well as the CRPC, for support under CCR–9120008, the Boeing Company, and the REDI Foundation. They also thank the Boeing Company for the hospitality that they were shown during the year of their visit.

Virginia Torczon and Michael Trosset wish to thank AFOSR and NSF for support under grants AFOSR–F49620–95–1–0210 and NSF CCR–9734044. They also received support under NASA Contract No. NAS1–19480 while they were in residence at the Institute for Computer Applications in Science and Engineering (ICASE), NASA Langley Research Center, Hampton, Virginia. They also thank the Boeing Company for the hospitality that they were shown during their visits as part of this collaboration.

References

[1] Barthelemy, J-F. M. and Haftka, R T. 1993: Approximation concepts for optimum structural design – a review. *Structural Optimization.* **5**, 129–144.

[2] Booker, A. J. 1994: DOE for computer output. Technical Report BCSTECH-94-052, Boeing Computer Services, Research and Technology, M/S 7L–68, Seattle, Washington 98124.

[3] Booker, A. J. 1996: Case studies in design and analysis of computer experiments. In *Proceedings of the Section on Physical and Engineering Sciences*, American Statistical Association.

[4] Booker, A. J.; Conn, A. R.; Dennis, J. E. Jr; Frank, P. D.; Trosset, M. W. and Torczon, V. 1995: Global modeling for optimization: Boeing/IBM/Rice collaborative project 1995 final report. Technical Report ISSTECH–95–032, Boeing Information & Support Services, Research and Technology, M/S 7L–68, Seattle, Washington 98124.

[5] Booker, A. J.; Dennis, J. E. Jr; Frank, P. D.; Serafini, D. B.; and Torczon, V. 1997: Optimization using surrogate objectives on a helicopter test example. Technical Report SSGTECH-97-027, Boeing Shared Services Group, Applied Research & Technology, M/S 7L–68, Seattle, Washington 98124. Also available as Technical Report 97-31, Department of Computational and Applied Mathematics, Rice University, Houston, Texas 77005-1892. To appear in Borggaard, J.; Burns, J.; Cliff, E. and Schreck, S., editors, *Optimal Design and Control.* Birkhauser, Cambridge, Massachusetts.

[6] Burgee, S. L.; Giunta, A. A.; Balabanov, V.; Grossman, B.; Mason, W. H.; Narducci, R.; Haftka, R. T. and Watson, L. T. 1996: A coarse-grained parallel variable-complexity multidisciplinary optimization paradigm. *Intl. J. Supercomputing Applications and High Performance Computing.* **10**(4), 269–299.

[7] Conn, A. R.; Scheinberg, K. and Toint, Ph. L. 1997: On the convergence of derivative-free methods for unconstrained optimization. In Iserles, A. and Buhmann, M., editors,

Approximation Theory and Optimization: Tributes to M. J. D. Powell, 83–108. Cambridge University Press, Cambridge, United Kingdom.

[8] Conn, A. R. and Toint, Ph. L. 1996: An algorithm using quadratic interpolation for unconstrained derivative free optimization. In Di Pillo, G. and Giannessi, F., editors, *Nonlinear Optimization and Applications*, 27–47. Plenum Publishing, New York.

[9] Cox, D. D. and John, S. 1997: SDO: a statistical method for global optimization. In Alexandrov, N. and Hussaini, M. Y., editors, *Multidisciplinary Design Optimization: State of the Art*, 315–329. SIAM, Philadelphia.

[10] Currin, C.; Mitchell, T.; Morris, M. and Ylvisaker, D. 1988: A Bayesian approach to the design and analysis of computer experiments. Technical Report ORNL–6498, Oak Ridge National Laboratory.

[11] Davis, C. 1954: Theory of positive linear dependence. *American Journal of Mathematics.* **76**, 733–746.

[12] De Boor, C. and Ron, A. 1992: Computational aspects of polynomial interpolation in several variables. *Mathematics of Computation.* **58**(198), 705–727.

[13] Dennis, J. E. Jr and Torczon, V. 1991: Direct search methods on parallel machines. *SIAM J. Optimization.* **1**(4), 448–474.

[14] Dennis, J. E. Jr and Torczon, V. 1997: Managing approximation models in optimization. In Alexandrov, N. and Hussaini, M. Y., editors, *Multidisciplinary Design Optimization: State-of-the-Art*, 330–347. SIAM, Philadelphia. Also available as Technical Report 95-19, Department of Computational and Applied Mathematics, Rice University, Houston, Texas 77005–1892.

[15] Dennis, J. E. Jr. and Walker, H. F. 1984: Inaccuracy in quasi-Newton methods: local improvement theorems. *Mathematical Programming Study.* **22**, 70–85.

[16] Dixon, L. C. W. and Szegö G. P., editors. 1978: *Towards Global Optimization 2.* North-Holland Pub. Co., Amsterdam.

[17] Efron, B. and Stein, C. 1981: The jackknife estimate of variance. *The Annals of Statistics.* **9**(3), 586–596.

[18] Frank, P. D. 1995: Global modeling for optimization. *SIAG/OPT Views-and-News.* **7**, 9–12.

[19] Geisser, S. and Eddy, W. F. 1979: A predictive approach to model selection. *Journal of the American Statistical Association.* **74**, 153–160.

[20] Giunta, A. A. 1997: *Aircraft Multidisciplinary Optimization using Design of Experiments Theory and Response Surface Modeling Methods*. PhD thesis, Virginia Tech. Available as MAD 97-05-01, May 1997, Department of Aerospace and Ocean Engineering, Virginia Tech, 215 Randolph Hall, Blacksburg, Virginia 24061.

[21] Jones, D. R.; Schonlau, M. and Welch, W. J. 1997: A data analytic approach to Bayesian global optimization. In *Proceedings of the ASA*.

[22] Koehler, J. R. and Owen, A. B. 1996: Computer experiments. In Ghosh, S. and Rao, C. R., editors, *Handbook of Statistics, Volume 13*, 261–308. Elsevier Science, New York.

[23] Levine, D. 1996: Users guide to the PGAPack parallel genetic algorithm library. Technical Report ANL-95/18, Argonne National Laboratory, 9700 South Cass Avenue, Argonne, Illinois 60439. Available from URL ftp://info.mcs.anl.gov/pub/pgapack/-pgapack.tar.Z.

[24] Lewis, R. M. and Torczon, V. 1996: Pattern search algorithms for bound constrained minimization. Technical Report 96–20, ICASE, NASA Langley Research Center, Hampton, Virginia 23681–2199. To appear in *SIAM J. Optimization*.

[25] Lewis, R. M. and Torczon, V. 1996: Rank ordering and positive bases in pattern search algorithms. Technical Report 96–71, ICASE, NASA Langley Research Center, Hampton, Virginia 23681–2199. In revision for *Mathematical Programming*.

[26] Lewis, R. M. and Torczon, V. 1997: Pattern search methods for linearly constrained minimization. Technical Report 98–03, ICASE, NASA Langley Research Center, Hampton, Virginia 23681–2199. To appear in *SIAM J. Optimization*.

[27] McKay, M. D.; Conover, W. J. and Beckman, R. J. 1979 A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*. **21**(2), 239–245.

[28] Owen, A. B. 1992: Orthogonal arrays for computer experiments, integration and visualization. *Statistica Sinica*. **2**, 439–452.

[29] Rinnooy Kan, A. H. G. and Timmer, G. T. 1984: A stochastic approach to global optimization. In Boggs, P. T.; Byrd, R. H. and Schnabel, R. B., editors, *Numerical Optimization 1984: Proceedings of the SIAM Conference on Numerical Optimization*, 245–262. SIAM, Philadelphia.

[30] Ripley, B. D. 1988: *Statistical Inference for Spatial Processes*. Cambridge University Press, Cambridge, United Kingdom.

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

[32] Serafini, D. B. 1998: A Framework for Managing Models in Nonlinear Optimization of Computationally Expensive Functions. Ph.D. Thesis, Rice University.

[33] Shultz, L. A.; Panda, B.; Tarzanin, F. J.; Derham, R. C.; Oh, B. K. and Dadone, L. 1994: Interdisciplinary analysis for advanced rotors - approach, capabilities and status. AHS Aeromechanics Specialists Conference, January 19–21, 1994, PS 4-1–4-15.

[34] Snir, M.; Otto, S. W.; Huss-Lederman, S.; Walker, D. W. and Dongarra, J. 1996: *MPI: The Complete Reference*. The MIT Press, Cambridge, Massachusetts.

[35] Stein, M. 1987: Large sample properties of simulations using latin hypercube sampling. *Technometrics*. **29**(2), 143–151.

[36] Tang, B. 1993: Orthogonal array-based latin hypercubes. *Journal American Statistical Association*. **88**(424), 1392–1397.

[37] Torczon, V. 1992: PDS: direct search methods for unconstrained optimization on either sequential or parallel machines. Technical Report 92–9, Department of Computational and Applied Mathematics, Rice University, Houston, Texas 77005–1892. In revision for *ACM Transactions on Mathematical Software*.

[38] Torczon, V. 1995: Pattern search methods for nonlinear optimization. *SIAG/OPT Views and News*. **6**, 7–11.

[39] Torczon, V. 1997: On the convergence of pattern search algorithms. *SIAM J. Optimization*. **7**(1), 1–25.

[40] Torczon, V. and Trosset, M. W. 1998: From evolutionary operation to parallel direct search: pattern search algorithms for numerical optimization. *Computing Science and Statistics*. **29**, 396–401.

[41] Trosset, M. W. and Torczon, V. 1997: Numerical optimization using computer experiments. Technical Report 97–38, ICASE, NASA Langley Research Center, Hampton, Virginia 23681–2199.

[42] Trosset, M. W. 1997: I know it when I see it: toward a definition of direct search methods. *SIAG/OPT Views and News*. **9**, 7–10.

[43] Watson, G. S. 1984: Smoothing and interpolation by kriging and with splines. *Mathematical Geology*. **16**, 601–615.