# An Efficient Class of Direct Search Surrogate Methods for Solving Expensive Optimization Problems with CPU-Time-Related Functions

Mark A. Abramson [*]   Thomas J. Asaki [†]   J. E. Dennis, Jr. [‡]

Raymond Magallanez [§]   Matthew Sottile [¶]

April 1, 2011

## Abstract

In this paper, we characterize a new class of computationally expensive optimization problems and introduce an approach for solving them. In this class of problems, objective function values may be directly related to the computational time required to obtain them, so that, as the optimal solution is approached, the computational time required to evaluate the objective is significantly less than at points farther away from the solution. This is motivated by an application in which each objective function evaluation requires both a numerical fluid dynamics simulation and an image registration process, and the goal is to find the parameter values of a predetermined reference image by comparing the flow dynamics from the numerical simulation and the reference image through the image comparison process. In designing an approach to numerically solve the more general class of problems in an efficient way, we make use of surrogates based on CPU times of previously evaluated points, rather than their function values, all within the search step framework of mesh adaptive direct search algorithms. Because of the expected positive correlation between function values and their CPU times, a time cutoff parameter is added to the objective function evaluation to allow its termination during the comparison process if the computational time exceeds a specified threshold. The approach was tested using the NOMADm and DACE MATLAB® software packages, and results are presented.

[*]The Boeing Company, PO Box 3707, MC 7L-21, Seattle, WA 98124 USA, `mark.a.abramson@boeing.com`.

[†]Washington State University, Department of Mathematics, PO Box 643113, Neill Hall 103, Pullman, WA 99164-3113 USA, `tasaki@wsu.edu`, `http://geometricanalysis.org/TomAsaki`.

[‡]Rice University, Department of Computational and Applied Mathematics, 8419 42nd Avenue SW, Seattle, WA 98136-2360 USA, `dennis@rice.edu`, `http://www.rice.edu/~dennis`.

[§]United States Air Force Academy, Department of Mathematical Sciences, USAFA, CO USA, `raymond.magallanez@usafa.edu`.

[¶]Galois, Inc., 421 SW 6th Ave., Suite 300, Portland, OR 97204, `matt@galois.com`.

1

# 1   Introduction

In this paper, we introduce a new class of optimization problems and a novel approach for numerically solving them. This class consists of minimizing an objective function $f : X \subset \mathbb{R}^n \to \mathbb{R}$ that is computationally expensive to evaluate, but becomes significantly less so as the solution is approached. That is, there is a reasonably strong correlation between objective function values and the CPU time required to compute them. We will refer to this class of problems as *CPU-time-related*. No assumption is made regarding the precise nature of the correlation, since it will generally not be known. (This is in contrast with work in which the correlation is measured; *e.g.*, see [34] and [27].) Typically, the objective function involves some type of engineering simulation or process modeling for which the computational time for simulation runs on a large set of problem instances is either infeasible (exceeds problem requirements) or impractical (months, years, or worse). While tackling the extreme computational requirements of large simulations is our ultimate goal, our primary motivation in this paper comes from an application whose objective function involves both a fluid dynamics simulation and an image registration process, the latter of which becomes much less expensive close to the solution. For this application, the feasible region $X$ is defined by simple bounds on the design variables, but the approach described here is sufficiently general to cover problems with general nonlinear constraints.

We note that other applications exist in which function evaluations are positively correlated with their required CPU times. In general, certain parameter estimation problems that require the numerical solution of differential equations at each evaluated point may require fewer iterations (and hence less CPU time) when the parameters are close to their true values. This property is also present in theater-level combat simulations, where one function evaluation represents a simulation given certain parameter settings. In the scenario of an enemy invasion, one common objective would be the employment of defensive forces in a way that minimizes enemy penetration (in distance). Enemy penetration is often highly correlated (though not perfectly) with the time it takes to stop the penetration. Since the expensive simulation would typically run until the enemy advance is halted (or shortly thereafter), the computational expense of an objective function evaluation (*i.e.*, one simulation run) is highly correlated with the objective function value.

The positive correlation between objective function value and the computational time leads us to a solution approach that integrates CPU runtime measures into the optimization process, allowing us to better utilize computational resources and significantly reduce computational time. Because of the computational expense of the function evaluations, our approach also involves the iterative use of surrogates. In this context, a surrogate

can be thought of as a much less expensive replacement for, but not necessarily a good approximation to, the objective function. In fact, in this paper we introduce surrogates based on CPU times in addition to those based on objective function values. An alternative approach is to make use of coarser representations of the simulation codes when further away from a solution [9]. The present work is independent of that course of action; *i.e.*, there is no reason why both ideas could not be implemented together (a topic for future research). For this reason, we fix the model fidelity here and confine our focus to exploiting the relationship between objective function value and CPU time; we hope to address model fidelity in a separate paper.

To maintain rigorous convergence properties of our approach, the use of surrogates is incorporated into the search step of the class of mesh adaptive direct search (MADS) algorithms. This is consistent with the surrogate management framework (SMF) introduced by Booker et al. [12]. This is an important distinction, as the poll step of the algorithm ensures convergence to a point satisfying certain necessary conditions for optimality, while the search step (which makes use of surrogates) is only used to make the process of convergence significantly more efficient for computationally expensive problems.

The remainder of the paper is as follows. In Section 2, we further motivate our work by discussing the details of our application. We present the MADS algorithm in Section 3 and discuss surrogates in more detail in Section 4, including some specific surrogate types and initialization strategies. In Section 5, we introduce new strategies for incorporating surrogates to efficiently solve our target class of problems. Numerical results on a specific instance of our application are given in Section 6, followed by some concluding remarks in Section 7.

## 2   An Applicable Class of Optimization Problems

The class of problems we target is motivated by an application in which each objective function evaluation requires both a fluid dynamics simulation and an image registration process. We consider the fluid dynamics application of discovering optimal simulation model parameters which most accurately reproduce a given experimentally obtained template image. Image registration is used to measure the degree to which any particular model output differs from this template data. For our initial study, we consider noisy template data constructed from simulation output.

The movement of fluids in a region $\Omega \subset \mathbb{R}^n, n \in \{2,3\}$ is governed by the well-known Navier-Stokes equations:

$$
\begin{aligned}
\frac{\partial}{\partial t} v + (v \cdot \nabla)v + \nabla p &= \frac{1}{Re} \triangle v + (1 - \beta \tilde{T})g, \\
\frac{\partial \tilde{T}}{\partial t} + v \cdot \nabla \tilde{T} &= \frac{1}{Re} \frac{1}{Pr} \triangle \tilde{T} + q''', \\
\text{div } v &= 0,
\end{aligned}
$$

3

where $u$ is a velocity field on $\mathbb{R}^n$, $p$ is the pressure field in $\Omega$, $g$ indicates body forces in $\Omega$, $Re \in \mathbb{R}$ is the Reynolds number of the flow, $Pr \in \mathbb{R}$ is Prandtl number of the flow, $\beta \in \mathbb{R}$ is the coefficient of thermal expansion, $q'''$ is the heat source, $\tilde{T}$ is the temperature, and $\triangle$ denotes the Laplace operator (sum of the unmixed second partial derivatives).

Our test example is that of the well-known *lid-driven cavity* problem [17], in which an initially stationary 2d fluid in a rectangular container is subject to forces imposed by the top boundary (lid) moving at a uniform horizontal velocity. This causes a circular pattern of flow to appear within the fluid over time. Since the Navier-Stokes equations cannot be solved analytically, they must be solved numerically using a finite element method and associated finite differencing scheme. We use the method of [17]. Figure 1(a) shows a snapshot of the fluid velocity at some positive time. Figure 1(b) shows the corresponding representation of the heat function $H$, which we will use as reference data. The heat function defines the two-dimensional heat flux $\Phi_q = \nabla \times H$, and is analogous to the hydrodynamic stream function which defines the (two-dimensional) velocity. For each combination of Reynolds number and simulation length, the velocity and viscosity of the fluid form a different circular heat flux pattern throughout the region. The goal will be to recover from the snapshot the (unknown) Reynolds number and simulation length.
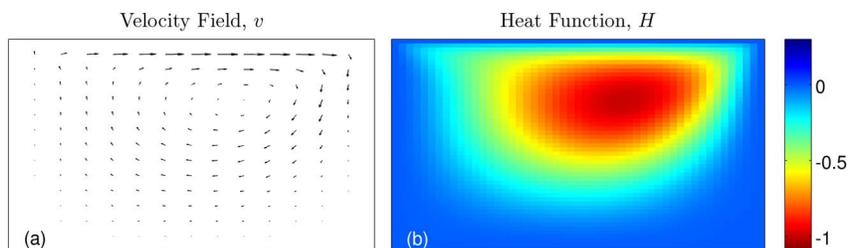


Figure 1: Example time snapshots of a lid-driven cavity simulation. The velocity field (a) at some time $t > 0$ shows a vortical structure. The heat function (b) at time $t$ serves as our example reference image.

Image registration is the process of estimating some optimal transformation $u^*$ between two images. Thus, a transformation $u$ is realized as a path through the space of images. A particular choice of $u$ will depend on the needs of the application. For example, in medical imaging it is desirable to compare images with minimal distortion, $\nabla \times u$. Other image comparison tasks benefit from minimizing the work required to "move" the intensities from one image to another. Different types of transformations are described in [30]. If we consider the classical inner product space $L_2(\Omega)$ of squared Lebesgue-integrable functions with its standard induced norm, a transformation of an image $T$ is given by $T_u(x) = T(x - u(x))$, where $u(x)$ is the displacement of the point $x$. The objective is to minimize the distance between a *reference image* $R$ and a *template image* $T$ through an

4

optimal warp transformation $T_u \in L_2(\Omega)$, as defined by some distance measurement $D$, and a smoothing or regularizing term $S$. This problem is given by

$$\min_u \; D[R, T_u] + \alpha S[u],$$ (1)

where $\alpha > 0$ governs the relative contributions of the two terms. We choose to illustrate our techniques using the curvature registration method of Fischer and Modersitzki [14], where

$$D[R, T_u] \;\; = \;\; \frac{1}{2}\|T_u - R\|_{L_2(\Omega)},$$ (2)

$$S[u] \;\; = \;\; \frac{1}{2}\sum_{i=1}^{n_d}\int_\Omega (\Delta u_i)^2 \, dx,$$ (3)

and $n_d$ is the dimension of the data space $\Omega$. Equation (3) can be viewed as an approximation to the curvature of the transformation $u$. The optimal transformation $u^*$ is the one that simultaneously minimizes image differences and transformation curvature.

The Euler-Lagrange equation for (1) is

$$(T_u(x) - R(x))\, \nabla T_u(x) + \alpha \Delta^2 u(x) = 0, \; x \in \Omega,$$ (4)

where we have applied Neumann boundary conditions $\nabla u_i = \nabla \Delta u_i = 0$, $i = 1, 2, \ldots, n_d$, on the boundary $\partial \Omega$ of $\Omega$. One method of solution is to take the associated time-dependent equation,

$$u_t(x, t) + \alpha \Delta^2 u(x, t) = (R(x) - T_u(x, t))\, \nabla T_u(x, t),$$ (5)

and compute the steady state solution $u_t^*(x, t) = 0$ using the iteration,

$$u^{(k+1)}(x, t) = u^{(k)}(x, t) + \tau u_t^{(k)}(x, t),$$ (6)

with appropriately chosen artificial time step $\tau$.

Figure 2 shows an example of an image registration for images of the temperature of two different parameterizations of a fluid flow simulation. The top left picture is the reference image $R$ for a specific set of (unknown) parameter values (e.g., Reynolds number of the fluid). For a different set of parameter values, the simulation is run, resulting in the template image $T$ shown in the top right picture. The image registration is then applied using (6) to solve (1)–(3). The resulting warped template image $T_u$ and the difference between $R$ and $T_u$ are shown in the bottom left and right images, respectively.

Given pixel points $\{x_i\}_{i=1}^{n_p} \subset \mathbb{R}^n$, where $n_p$ is the number of pixels in the image, the objective measure of the goodness of simulation parameter choices is given by the optimal transformation; namely,

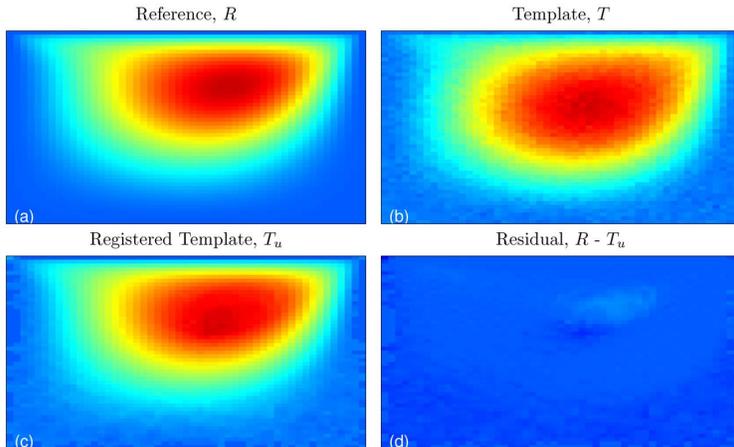$$f = \|u^* - \bar{u}\|_2, \quad \bar{u} = \frac{1}{n_p}\sum_{i=1}^{n_p} u^*(x_i)$$ (7)

5

Figure 2: Image registration example. The noisy intensities of the template image (b) are transformed into a registered image (c) intended to match a reference image (a). The optimal transformation $u^*$ is determined by (1). The residual $R - T_u$ is shown in (d).

The subtraction of the means in (7) allows for a zero objective function value for images that are identical except for translational alignment issues. If the images are very similar, the numerical image registration scheme (6) requires only a few iterations to transform $T$ into $R$, resulting in less computational time as well as a small distance value. On the other hand, images that are relatively dissimilar require more iterations, increasing the computational time. Thus, the objective function value $f$ and its associated computational time are expected to be strongly correlated. We should also point out that the computational cost of an image registration is not trivial. When image deformation methods are applied instead of simpler affine transformations, the cost due to the additional computational workload can be significant.

# 3    Mesh Adaptive Direct Search (MADS)

The class of mesh adaptive direct search (MADS) algorithms was introduced by Audet and Dennis [8] as a way of extending generalized pattern search (GPS) [6, 22, 23, 40] to optimization problems with nonlinear constraints without the use of penalty parameters [24] and with a stronger theoretical justification than is available with filters [7]. Each iteration of GPS and MADS algorithms consists of a search and a poll step performed on a mesh formed by a set of $n_D$ directions $D$ that positively spans $\mathbb{R}^n$. These algorithms are applied not to the objective function $f$, but to the barrier objective $f_X \equiv f + \psi_f$, where $\psi_f$ is the indicator function for $f$; it is zero at $x \in X$ and $\infty$ for $x \notin X$.

6

The mesh at iteration $k$, which is not actually constructed, can be expressed as

$$M_k = \bigcup_{x \in S_k} \{x + \Delta_k^m Dz : z \in \mathbb{N}^{n_D}\},$$

where $S_k$ is the finite set of points where the objective function $f$ had been evaluated by the start of iteration $k$ (so $S_0$ is the set of initial feasible points), and the *mesh size parameter* $\Delta_k^m$ controls how coarse or fine the mesh is.

The search step is very flexible, as it consists of evaluating $f_X$ at any finite number of mesh points. One could choose to sample randomly, sample at points generated by an experimental design, run a few iterations of a favorite heuristic, such as a genetic algorithm, or simply do nothing. For computationally expensive functions, the search step usually consists of constructing (or recalibrating) surrogate functions and solving a surrogate optimization problem on the mesh. The resulting solution, along with any other promising points, is then evaluated by $f_X$.

If the search step fails to yield a mesh point with lower objective function value, the poll step is performed. It consists of evaluating the set of adjacent mesh points $P_k$, called the *poll set*; namely,

$$P_k = \{x_k + \Delta_k^m d : d \in D_k\} \subset M_k,$$

where the current iterate $x_k$ is called the *frame center*, and $D_k$ is a positive spanning set of directions satisfying certain properties that will ensure appropriate convergence properties of the algorithm. In GPS methods, the condition of $D_k \subset D$ must hold at each iteration, and $D_k$ is chosen to include directions that conform to the boundary of any nearby constraint [23]. This is sufficient for problems with a finite number of linear constraints because we include in $D$ all possible conforming directions at any point in the feasible region. However, for more general problems with nonlinear constraints, this condition is insufficient to ensure convergence.

In the more general class of MADS algorithms, a new parameter $\Delta_k^p$, called the *poll size parameter* is introduced, which satisfies $\Delta_k^m \leq \Delta_k^p$ for all $k$, and

$$\lim_{k \in K} \Delta_k^m = 0 \Leftrightarrow \lim_{k \in K} \Delta_k^p = 0 \text{ for any infinite subset of indices } K. \tag{8}$$

Under this construction, GPS now becomes the specific instance of MADS with $\Delta_k = \Delta_k^m = \Delta_k^p$. In MADS, $D_k$ is not a subset of $D$, but instead must satisfy the following properties:

- Each nonzero $d \in D_k$ can be written as a nonnegative integer combination of the directions in $D$; *i.e.*, $d = Du$ for some vector $u \in \mathbb{N}^{n_D}$ that may depend on the iteration number $k$.

- The distance from the frame center $x_k$ to a poll point $x_k + \Delta_k^m d$ is bounded by a constant times the poll size parameter; *i.e.*, $\Delta_k^m \leq \|d\| \leq \Delta_k^p \max\{\|d'\| : d' \in D\}$.

- Limits (as defined in Coope and Price [13]) of the normalized sets $D_k$ are positive spanning sets.

The idea in MADS is that $\Delta_k^m$ approaches zero faster than $\Delta_k^p$, which increases the number of possible directions from which to select for inclusion in $D_k$. This is illustrated in Figure 3, where GPS and MADS frames (in rows 1 and 2, respectively), constructed from the standard set of $2n$ positive and negative standard coordinate directions, are depicted in two dimensions. In each case, the thick-lined box is called a *frame* (with frame center $x_k$), and the points where it intersects the mesh are at a relative distance of $\Delta_k^p$ from the frame center $x_k$. In MADS, any set of positive spanning directions that yields mesh points inside the frame (*e.g.*, $p_1$, $p_2$, and $p_3$) may be chosen.
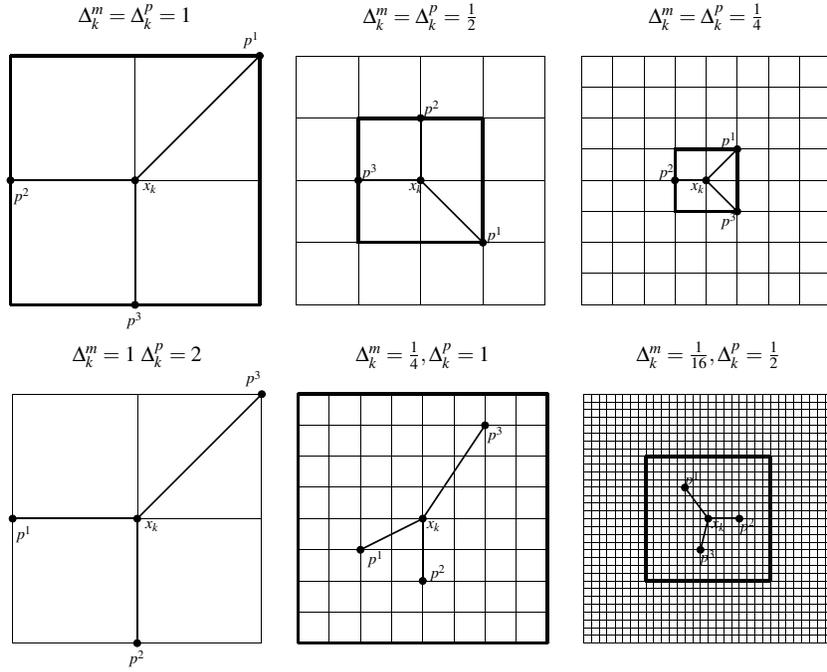


Figure 3: GPS and MADS frames in $\mathbb{R}^2$

If either the search or poll succeeds in finding an improved mesh point, it becomes the new iterate $x_{k+1} \in X$, and the mesh size is retained or increased. If neither step succeeds, then the current iterate is retained and the mesh size is reduced. More specifically, given a fixed rational number $\tau > 1$ and two integers $w^- \leq -1$ and $w^+ \geq 0$, $\Delta_k^m$ is updated

according to the rule

$$\Delta_{k+1}^m = \tau^{w_k} \Delta_k^m$$

$$\text{for some } w_k \in \left\{ \begin{array}{ll} \{0, 1, \ldots, w^+\} & \text{if an improved mesh point is found,} \\ \{w^-, w^- + 1, \ldots, -1\} & \text{otherwise.} \end{array} \right. \quad (9)$$

A general MADS algorithm is given in Figure 4.

---

0. **Initialization**: Let $x_0 \in X$, set $\Delta_0^p \geq \Delta_0^m > 0$. Set $k = 0$.

1. **Search:** Evaluate $f_X$ at a finite subset of points on the mesh $M_k$.

2. **Poll:** If the search step fails, evaluate $f_X$ at points in the frame $P_k$.

3. **Update:** If the search or poll finds a new improved mesh point, set it to be $x_{k+1}$; otherwise, set $x_{k+1} = x_k$. Update $\Delta_{k+1}^m$ according to (9), and $\Delta_{k+1}^p$ according to (8). Increment $k \leftarrow k + 1$ and go to Step 1.

---

Figure 4: A general MADS algorithm

Convergence of MADS depends on selecting directions so that the union of normalized poll directions used becomes asymptotically dense on the unit sphere [8]. Specific instances of MADS have been shown to achieve this condition with probability one (by randomly selecting $D_k$) [8] and deterministically (through the use of Halton sequences) [4].

More details, including proofs of convergence to appropriately defined first-order stationary points (even for non-smooth functions), are given for GPS and MADS in [6] and [8], respectively. Second-order stationarity results for GPS and MADS are studied in [2] and [3], respectively.

## 4    Surrogate Functions

The idea for surrogates first appeared as "approximation concepts" in the work of Schmit and Miura [21]. Booker et al. [12] characterize a class of problems for which surrogate functions would be an appropriate approach, suggest a surrogate composition, and set forth a general Surrogate Management Framework (SMF) for using surrogates to numerically solve optimization problems. More information about surrogate-based optimization can be found in [15], [41], and [45].

Most surrogates are one of two types: simplified physics or response-based. A simplified physics model, also known as a low-fidelity model, makes certain physical assumptions that significantly reduce the computational cost by eliminating complex equations and even the number of variables. Although several novel approaches exist in the literature for treating

this class of surrogates (e.g., see [5] or [33]), the actual construction of the models is problem-dependent.

To handle our target class of problems, we use as surrogates a class of response-based kriging approximation models [35] (see also, [20], [29], and [38]). These are not the only response-based models, and in fact, there have even been cases where multiple surrogate types are applied to the same data [16, 42, 43, 44] in an effort to improve performance.

Given a set of known data points (or sites) $\{s_i\}_{i=1}^{n_s} \subset \mathbb{R}^n$ and their deterministic response or function values $y_s \in \mathbb{R}^{n_s}$ (*i.e.*, $[y_s]_i = y(s_i)$ for $i = 1, 2, \ldots, n_s$), the deterministic function $\hat{y}(z)$ is modeled as a realization of a stochastic process,

$$Y(z) = \sum_{j=1}^{p} \beta_j \phi_j(z) + Z(z) = \beta^T \phi(z) + Z(z),$$

where $Y(z)$ is the sum of a regression model with coefficients $\beta = [\beta_1, \beta_2, \ldots, \beta_p] \in \mathbb{R}^p$ and basis functions $\phi = [\phi_1, \phi_2, \ldots, \phi_p]$, and a random variable $Z(z)$, where $Z : \mathbb{R}^n \longrightarrow \mathbb{R}$, with mean zero and covariance $V(w, z) = \sigma^2 R(\theta, w, z)$ between $Z(w)$ and $Z(z)$, $\sigma^2$ is the process variance, and $R(\theta, w, z)$ is the correlation function of $w$ and $z$. The parameter $\theta \in \mathbb{R}^n$ controls the shape of the correlation function. Kriging produces an approximate function value at an unknown point $s_0 \in \mathbb{R}^n$ using weights on known responses; namely,

$$\hat{y}(s_0) = c(s_0)^T y_s,$$

where $c(s_0) \in \mathbb{R}^{n_s}$ is a vector of weights. Details for computing $c(s_0)$ are given in [25] or [35], for example.

The optimization process requires the computation of $R(\theta, x, x)^{-1}$. However, this is difficult numerically because $R$ can become ill-conditioned as points cluster together during the convergence process of MADS [10]. The increase in the condition number of $R(\theta, x, x)$ (which can be estimated and monitored) can greatly impact the computed value of $\theta$ or prevent the calculation of the regression coefficients altogether. Booker [10] alleviates this problem in practice by introducing a second correlation function for all the points that are generated after the initial surrogate is formed. He also suggests the use of additional correlation functions if the ill-conditioning recurs. An alternative approach for alleviating numerical instability due to the clustering of points was studied in [28]. For problems with CPU-time-related functions, since the clustering of points coincides with smaller CPU times, we can also choose to simply skip the search step whenever an estimate of the condition number (based on [18, 19]) of the kriging correlation matrix becomes too large.

## 5   New Surrogate Strategies

In this section, we introduce a new search step for handling CPU-time-related functions. First, if objective function values and CPU times are positively correlated, then improvement in the objective function should not be expected once the computational time exceeds

a certain amount. To avoid wasting unnecessary CPU time, we introduce a CPU time cut-off parameter $t_k^{cut} > 0$ to allow a function evaluation to be aborted if it is taking too long to perform. A value of $t_k^{cut} = \infty$ means that the function is evaluated normally without being aborted.

At each point $x \in X$ that is evaluated by $f$, we record its function value $z = f(x)$ and the CPU time $t = t(x)$ required to evaluate $f(x)$. For a specified time cutoff parameter value of $t_k^{cut}$, we can represent a function evaluation by $[z, t] = f(x, t_k^{cut})$. Once the time for computing the function value exceeds the value specified by $t_k^{cut}$, evaluation of $f(x)$ is aborted without returning a value for $z$ (or $z$ is set to be infinity or an arbitrarily large number) and with $t$ set to $t_k^{cut}$. One possible approach we considered is to set $t_{k+1}^{cut} = \alpha t_k$, $\alpha \geq 1$, where $t_k$ is the recorded CPU time for the current best iterate $x_k$ (*i.e.*, in our notation, $[z_k, t_k] = f(x_k, t_k^{cut})$).

The CPU time relation also means that a surrogate based on either the objective function values or CPU times would probably be a good predictor of decrease in the objective. In fact, a surrogate on the CPU time has the added advantage that it always returns a value, whereas, the objective function would be aborted if $t_k^{cut}$ is exceeded at iteration $k$. We denote these surrogates on objective values and CPU times by $\hat{f}_k(\cdot)$ and $\hat{t}_k(\cdot)$ (at iteration $k$), respectively, and we consider the following four surrogate optimization problems, whose solutions we would expect to be good subsequent trial points for the true objective function:

$$\min_{x \in X} \ \hat{f}_k(x), \tag{10}$$

$$\min_{x \in X} \ \hat{t}_k(x), \tag{11}$$

$$\min_{x \in X} \ \hat{f}_k(x), \quad \text{s.t.} \quad \hat{t}_k(x) \leq t_k^{cut} + \varepsilon, \tag{12}$$

$$\min_{x \in X} \ \hat{t}_k(x), \quad \text{s.t.} \quad \hat{f}_k(x) \leq z_k. \tag{13}$$

The parameter $\varepsilon > 0$ added to the constraint in (12) is a small constant offset to allow for variability in computational time. We do this to prevent the situation where unfortunate variations in CPU time result in feasible points that are flagged as infeasible with respect to 12.

We anticipate that the mount of correlation between function values and CPU times may be different for different problems. If the correlation is not as strong, then the time-based surrogates will not predict improvement as well. From the standpoint of convergence, the MADS poll step overcomes poor correlation, but at a higher cost. In fact, the class of MADS algorithms has been shown to be robust in solving problems, even when the surrogate predicts poorly [11].

The surrogate optimization problem is typically solved using a recursive call to MADS (though theoretically, the surrogate optimization problem could be solved by many optimization codes). Constraints in (12)–(13) are treated by the extreme barrier approach of

only allowing feasible points (or setting the function value at any infeasible trial point to infinity).

We should note that the combination of MADS with a barrier and the use of the $t_k^{cut}$ in the original optimization problem causes a dilemma when using surrogate functions. Using the parameter $t_k^{cut}$ to stop unprofitable function evaluations is good for saving computational time, but it results in infinite or arbitrarily large function values, which cannot be used to construct a good surrogate. To overcome this, we simply set the function value to the largest value seen thus far in the iteration process, whenever the time cutoff threshold is exceeded. Our algorithm can be summarized as simply MADS with the specific $k$th search step given in Figure 5.

---

- For all previously evaluated points $X_k$, construct surrogate functions $\hat{f}_k(\cdot)$ or $\hat{t}_k(\cdot)$.

- Solve a surrogate problem (one of (10)–(13)), yielding a set of trial points $S_k$.

- Evaluate points $x \in S_k$ using $[z,t] = f(x, t_k^{cut})$ until an improved mesh point has been found, or until all points in $S_k$ have been evaluated.

- If $z < z_k$ for some $x \in S_k$,
    Set $x_{k+1} = x,\ z_{k+1} = z,\ t_{k+1} = t,$ and update $t_{k+1}^{cut}$.
    Set $k = k + 1$ and repeat search
  End

---

Figure 5: MADS search step $k$ for CPU-time-related functions

## 6 A Numerical Example

We now present some numerical results from a specific example of the class of problems described in the Section 2, the *lid-driven cavity* problem [17]. Additional results and discussion are given in [26]. At one particular Reynolds number and simulation length, a reference image of the heat flux pattern is captured and then noise is added to the image, so as to represent what one might to see in experimentally obtained physical measurements. As stated in Section 2, the goal is to run a simulation for different Reynolds numbers and simulation lengths, capture the template image, and compare the template and reference images of heat in an attempt to determine the original Reynolds number and simulation length set for the reference image. Figure 2 shown earlier actually shows reference and template images for this very problem.

Since this problem has only bound constraints, we applied GPS with the search step described by Figure 5. To do this, we used two MATLAB® software packages, NOMADm

[1] for the implementation of MADS/GPS, and DACE [31] to build the kriging surrogates (with some custom-built files) for the search step. The use of kriging functions as surrogates requires specification of the data sites, and regression and correlation functions. For constructing an initial surrogate, the set of initial data sites can be chosen via experimental design [36] or by sampling a set of "space-filling" points, such as Latin hypercube designs [37, 39] or orthogonal arrays [32]. Experimental designs generally require more function evaluations, especially in larger dimensions. However, since our problem is of small dimension, we chose to use a 9-point central composite design (CCD). This also allowed us to use a second-order regression function for the kriging model, which we found to be more accurate than lower-order functions.

The surrogate optimization problem was solved by a recursive call to MADS from within the search step. An estimate of the condition number of the kriging correlation matrix was monitored using the MATLAB `condest` command, which is based on [18, 19]. The bounds on Reynolds number and simulation length (in seconds) were set to be $[0, 5000]$ and $[3, 8]$, respectively.

For each scenario, different variations of the algorithm are applied and compared to a base case. The base case implementation uses GPS with an empty search step (*i.e.*, it is skipped), a single initial point or set of CCD points, and $t_k^{cut} = \infty$ for all $k$. This allows for a full evaluation of all points and a comparative analysis of the proposed algorithm. The other cases use the partial and full implementation of the search step presented in Figure 5.

We first made some preliminary runs using a strategy of setting $t_{k+1}^{cut} = t_k$ at each iteration. However, these runs turned out to be unsuccessful in forming good surrogates (*i.e.*, surrogates that routinely found good trial points to evaluate) because our main assumption of CPU time correlation turned out not to hold in certain parts of the domain. If the template image is too dissimilar from the reference image, the image registration process actually terminates prematurely – with a lower CPU time and a much worse objective function value. This is seen in Figure 6, which shows the image registration time and objective function value for each trial point evaluated in the subsequent run, in which we set $t_{k+1}^{cut} = 2t_k$ at each iteration. This choice seemed to rectify the situation. Note that we only include image registration time here because the simulation time was roughly constant over all evaluated points, and for this particular problem, the image registration time was actually the more dominant time.

We also experienced ill-conditioning of the correlation matrix as a solution is approached, which is caused by trial points becoming more clustered together. This was remedied by invoking an empty search (*i.e.*, not optimizing the surrogate) whenever the matrix became ill-conditioned. This is not unreasonable in this case because the CPU time correlation means that function values are probably much less expensive at this point in the iteration process, and the use of surrogates is then not as important.

The results for each case are shown in Table 1, where the column headings denote, respectively, the type of search step executed (Search), type of initial points used ($x_0$), final
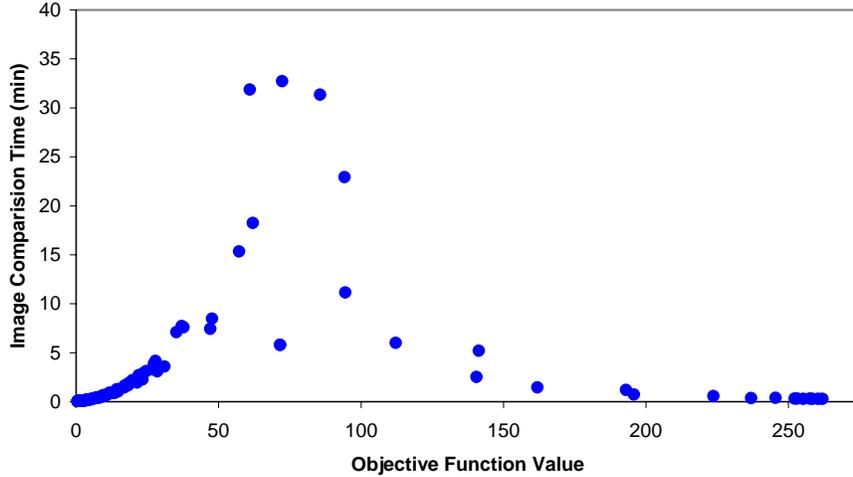
Figure 6: Time correlation for the lid-driven cavity problem

solution ($x^*$), number of iterations (nIter), number of function evaluations (nFunc), CPU minutes required (CPU), and the ratio of successful to total surrogate search steps executed (Successes). For the latter, a success refers to a surrogate optimizer that actually improves the objective function (as opposed to one that does not, due to poor prediction of the surrogate). Except for the "None" designation, the search types refer to the four surrogate strategies shown in (10)–(13). The first letter indicates the objective function ($\hat{f}(x)$ vs. $\hat{t}(x)$), and the second (if present) indicates the constraint. The initial point types consist of using a single initial point in the geometric center of the bound constrained feasible region (center), a randomly chosen initial feasible point (random), or central composite design (CCD). The final solution is expressed as [Reynolds number, simulation length (seconds)]. The first three runs are base cases with no search step or time cutoff parameter used, while the last seven cases are different variations of the new search step, the first three being similar to the base case (no search step) except for the use of the time cutoff parameter to abort expensive function evaluations. (The random point is the same random initial point that was used for the base case.) The final four cases employ one of the surrogate optimization problems (10)–(13), as just described.

All runs successfully found the optimal solution at essentially the same parameter values (with $f(x*) = 0.60$ in all cases). As hoped for, the CPU time was significantly lower for the two time-based surrogates ((11) and (13)) than all the other cases, despite costing more function evaluations than many other cases, including all three base cases. When using the $t_k^{cut}$ parameter and a CCD design, the time-based surrogates achieved a 32.3% reduction in CPU time. Since the time-based surrogates achieved this despite more function evaluations,

14

Table 1: GPS: Lid-Driven Cavity Results

Full-Time ($t_k^{cut} = +\infty$):

| Search | $x_0$ | $x^*$ | nIter | nFunc | CPU (min) | Successes |
|--------|-------|-------|-------|-------|-----------|-----------|
| None | center | [134.13, 4.76] | 56 | 123 | 126.73 | |
| None | random | [134.12, 4.76] | 58 | 118 | 178.31 | |
| None | CCD | [134.13, 4.76] | 40 | 107 | 196.58 | |

Cut-Time ($t_k^{cut} = 2 \times t_k$):

| Search | $x_0$ | $x^*$ | nIter | nFunc | CPU (min) | Successes |
|--------|-------|-------|-------|-------|-----------|-----------|
| None | center | [134.13, 4.76] | 80 | 157 | 257.67 | |
| None | random | [134.12, 4.76] | 92 | 162 | 796.40 | |
| None | CCD | [134.13, 4.76] | 40 | 107 | 109.73 | |
| $\hat{f}$ s.t. $\hat{t}$ | CCD | [134.19, 4.76] | 73 | 162 | 135.78 | 23/44 |
| $\hat{f}$ | CCD | [134.19, 4.76] | 72 | 165 | 182.89 | 15/34 |
| $\hat{t}$ s.t. $\hat{f}$ | CCD | [134.19, 4.76] | 52 | 133 | 74.00 | 7/15 |
| $\hat{t}$ | CCD | [134.19, 4.76] | 49 | 127 | 74.48 | 5/13 |

this also indicates that for this class of problems, the number of function evaluations is not a good measure of the efficiency of the different implementations.

The results with no search illustrate the importance of finding an appropriate initial point to set up the time cutoff parameter. With only one initial point, this parameter needs several iterations to build enough slack to allow the sequence of points to overcome the local nature of the CPU time correlation. The extra iterations resulted in a different path to a solution, which required significantly more CPU time. However, when using a initial CCD (with an empty search step), the results are identical except for the CPU time. In this case, using the time cutoff parameter saved almost 90 CPU minutes.

Figure 7 further illustrates the performance of the four surrogate strategies, with each color representing a different strategy (corresponding to (10)–(13)), and each shape (diamond or square) representing the source (search or poll step, respectively) of the improvement. The figure shows that the surrogates based on computational time achieve lower function values quicker than the surrogates based on function values. Furthermore, regardless of the surrogate objective, the use of a constraint (see (12) and (13)) in each case resulted in faster initial convergence than the corresponding unconstrained version ((10) and (11), respectively).
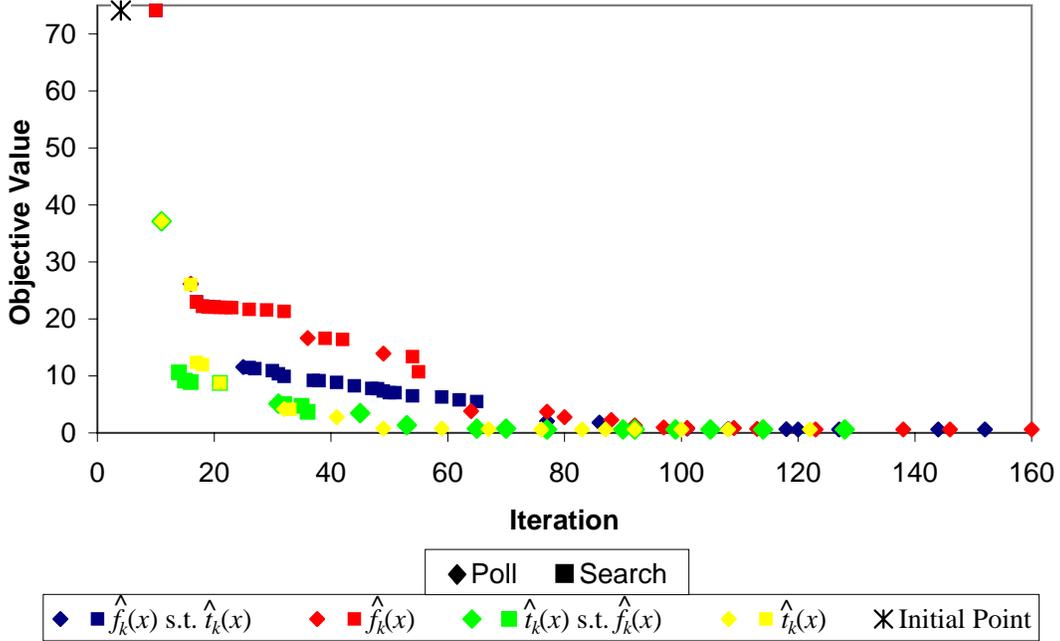
Figure 7: Decreasing Function Value

# 7 Concluding Remarks

This paper represents a first attempt at numerically solving the challenging class of optimization problems in which function values and the CPU times required to compute them are correlated. Exploiting acquired knowledge about the relationship between objective function values and their CPU times appears to be a useful and efficient means of solving this class of problems. One challenge is dealing with the extent to which the CPU time correlation property holds in practice, which may not be fully understood. The implementation of the time cutoff parameter was a useful way to reduce the time required to find a numerical solution. However, while it can be used to stop the image registration algorithm, it cannot stop the numerical simulation, since the image registration requires the image obtained from the full simulation.

Since $t_k^{cut}$ is controlled by the user, one potential improvement would be a more systematic approach to updating it, rather than the trial-and-error approach used here. Since function values and computational times are stored in order to construct surrogates, they can also be used to measure the correlation between the two. Higher values of $t_k^{cut}$ can be assigned whenever the correlation is low, and vice versa.

16

Not using the surrogates when ill-conditioning occurs was a simple tactic that made sense for the particular problem we solved. However, a more effective means to combat this problem may be the use of a trust region (e.g., see [5]), both to constrain the optimization of the surrogate and to screen out points used in the construction of the surrogate. The size of the trust region could be based on the frame or mesh size parameter.

## Acknowledgments

## References

[1] M. A. Abramson. NOMADm optimization software. http://www.gerad.ca/NOMAD/-Abramson/NOMADm.html.

[2] M. A. Abramson. Second-order behavior of pattern search. *SIAM J. Optim.*, 16(2):315–330, 2005.

[3] M. A. Abramson and C. Audet. Second-order convergence of mesh adaptive direct search. *SIAM J. Optim.*, 17(2):606–619, 2006.

[4] M. A. Abramson, C. Audet, J. E. Dennis, Jr., and S. Le Digabel. Orthomads: A deterministic instance with orthogonal directions. *SIAM J. Optim.*, 20(2):948–966, 2009.

[5] N. Alexandrov, J. E. Dennis, Jr., R. Lewis, and V. Torczon. A trust region framework for managing the use of approximation models in optimization. *Struct. Optim.*, 15:16–23, 1998.

[6] C. Audet and J. E. Dennis, Jr. Analysis of generalized pattern searches. *SIAM J. Optim.*, 13(3):889–903, 2003.

[7] C. Audet and J. E. Dennis, Jr. A pattern search filter method for nonlinear programming without derivatives. *SIAM J. Optim.*, 14(4):980–1010, 2004.

[8] C. Audet and J. E. Dennis, Jr. Mesh adaptive direct search algorithms for constrained optimization. *SIAM J. Optim.*, 17(2):188–217, 2006.

[9] D. Bethea. Improving mixed variable optimization of computational and model parameters using multiple surrogate functions. Master's thesis, Graduate School of Engineering and Management, Air Force Institute of Technology, Wright-Patterson AFB, OH, 2008.

[10] A. J. Booker. Well-conditioned Kriging models for optimization of computer simulations. Technical Report M&CT-TECH-00-002, Boeing Computer Services, Research and Technology, M/S 7L–68, Seattle, Washington 98124, 2000.

[11] A. J. Booker, J. E. Dennis, Jr., P. D. Frank, D. W. Moore, and D. B. Serafini. Managing surrogate objectives to optimize a helicopter rotor design – further experiments. AIAA Paper 1998–4717, Presented at the 8th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, St. Louis, 1998.

[12] A. J. Booker, J. E. Dennis, Jr., P. D. Frank, D. B. Serafini, V. Torczon, and M. W. Trosset. A rigorous framework for optimization of expensive functions by surrogates. *Struct. Optim.*, 17(1):1–13, February 1999.

[13] I. D. Coope and C. J. Price. A direct search conjugate directions algorithm for unconstrained minimization. *Australian & New Zealand Industrial and Applied Mathematics (ANZIAM) Journal*, 42, electronic Part C:C478–498, November 2000.

[14] B. Fischer and J. Modersitzki. Curvature based image registration. *J. Math. Imaging Vision*, 18(1):81–85, 2003.

[15] A. I. J. Forrester and A. J. Keane. Recent advances in surrogate-based optimization in aerospace sciences. *Progress in Aerospace Sciences*, 45(1-3):50–79, 2009.

[16] D. Ginsbourger, C. Helbert, and L. Carraro. Discrete mixtures of kernels for kriging-based optimization. *Quality and Reliability Engineering International*, 24(6):681–691, 2008.

[17] M. Griebel, T. Dornseifer, and T. Neunhoeffer. *Numerical Simulation in Fluid Dynamics: a Practical Introduction*. SIAM, New York, 1998.

[18] W. W. Hager. Condition estimates. *SIAM J. Sci. Stat. Comput.*, 5(2):311–316, 1984.

[19] N. J. Higham and F. Tisseur. A block algorithm for matrix 1-norm estimation with an application to 1-norm pseudospectra. *SIAM J. Matrix Anal. Appl.*, 21(4):1185–1201, 2000.

[20] J. P. C. Kleijnen. Kriging metamodeling in simulation: a review. *European Journal of Operational Research*, 192(3):707–716, 2009.

[21] Jr. L. A. Schmit and H. Miura. Approximation concepts for efficient structural synthesis. Technical Report CR-2552, NASA, 1976.

[22] R. M. Lewis and V. Torczon. Pattern search algorithms for bound constrained minimization. *SIAM J. Optim.*, 9(4):1082–1099, 1999.

[23] R. M. Lewis and V. Torczon. Pattern search methods for linearly constrained minimization. *SIAM J. Optim.*, 10(3):917–941, 2000.

[24] R. M. Lewis and V. Torczon. A globally convergent augmented Lagrangian pattern search algorithm for optimization with general constraints and simple bounds. *SIAM J. Optim.*, 12(4):1075–1089, 2002.

[25] S. N. Lophaven, H. B. Nielsen, and J. Søndergaard. Aspects of the MATLAB toolbox DACE. Technical Report IMM-TR-2002-13, Technical University of Denmark, Copenhagen, 2002.

[26] R. Magallanez. Surrogate strategies for computationally expensive optimization problems with cpu-time correlated functions. Master's thesis, Graduate School of Engineering and Management, Air Force Institute of Technology, 2007.

[27] V. E. Marin, J. A. Rincon, and D. A. Romero. A comparison of metamodel-assisted prescreening criteria for multi-objective genetic algorithms. In *ASME 2009 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference*, San Diego, CA, Aug 30-Sep 2 2009. DETC2009-87736.

[28] J. D. Martin. Robust kriging models. In *51th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, Orlando, FL, April 12-15 2010. AIAA 2010-2854.

[29] J. D. Martin and T. W. Simpson. Use of kriging models to approximate deterministic computer models. *AIAA Journal*, 43(4):853–863, 2005.

[30] J. Modersitzki. *Numerical Methods for Image Registration*. Oxford University Press, 2004.

[31] H. B. Nielsen. DACE surrogate models. http://www2.imm.dtu.dk hbn/dace.

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

[33] T. D. Robinson, M. S. Eldred, K. E. Willcox, and R. Haimes. Strategies for multifidelity optimization with variable dimernsional hierarchical models. In *Proc. 47th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conf. (2nd AIAA Multidisciplinary Design Optimization Specialist Conf.)*, Newport, Rhode Island, May 2006.

[34] D. A. Romero, C.H. Amon, and S. Finger. A study of covariance functions for multi-response metamodeling for simulation-based design and optimization. In *ASME 2008 International Design Engineering Technical Conference*, Brooklyn, NY, August 3-6 2008. DETC2008-50061.

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

[36] T. J. Santner, B. J. Williams, and W. I. Notz. *The Design and Analysis of Computer Experiments*. Springer Verlag, 2003.

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

[38] M. L. Stein. *Interpolation of spatial data: some theory for kriging*. Springer, 1999.

[39] B. Tang. Orthogonal array-based Latin hypercubes. *J. Amer. Statist. Assoc.*, 88(424):1392–1397, 1993.

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

[41] F. A. C. Viana, C. Gogu, and R. T. Haftka. Making the most out of surrogate models: tricks of the trade. In *ASME 2010 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference*, Montreal, Canada, August 16-18 2010. DETC2010-28813.

[42] F. A. C. Viana, R. T. Haftka, and L. T. Watson. Why not run the efficient global optimization algorithm with multiple surrogates? In *51th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, Orlando, FL, April 12-15 2010. AIAA 2010-3090.

[43] F.A.C. Viana and R.T. Haftka. Using multiple surrogates for metamodeling. In *7th ASMO-UK/ISSMO International Conference on Engineering Design Optimization*, Bath, UK, July 7-8 2008.

[44] I. Voutchkov and A. J. Keane. Multiobjective optimization using surrogates. In *7th International Conference on Adaptive Computing in Design and Manufacture*, pages 167–175, Bristol, UK, 2006.

[45] G. G. Wang and S. Shan. Review of metamodeling techniques in support of engineering design optimization. *Journal of Mechanical Design*, 129(4):370–380, 2007.