Bayesian Guided Pattern Search for Robust Local Optimization

Matthew Taddy, Herbert K. H. Lee, Genetha A. Gray, and Joshua D. Griffin[*]

February 12, 2009

Abstract

Optimization for complex systems in engineering often involves the use of expensive computer simulation. By combining statistical emulation using treed Gaussian processes with pattern search optimization, we are able to perform robust local optimization more efficiently and effectively than using either method alone. Our approach is based on the augmentation of local search patterns with location sets generated through improvement prediction over the input space. We further develop a computational framework for asynchronous parallel implementation of the optimization algorithm. We demonstrate our methods on two standard test problems and our motivating example of calibrating a circuit device simulator.

KEY WORDS: robust local optimization; improvement statistics; response surface methodology; treed Gaussian processes.

[*]M. Taddy is an Assistant Professor at the University of Chicago Booth School of Business, 5807 S. Woodlawn Ave, Chicago, IL 60637. H. K. H. Lee is a Professor, Department of Applied Mathematics and Statistics, University of California, Santa Cruz. G. A. Gray is a technical staff member at Sandia National Laboratories, Livermore, California. J. D. Griffin is a researcher staff member at the SAS Institute. This work was partially supported by Sandia grants 496420 and 673400, and NSF grants DMS-0112069 and DMS-0504851. Sandia National Laboratories is a multi-program laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under contract DE-AC04-94AL85000. Any opinions, findings, and conclusions expressed in this material are those of the authors and do not necessarily reflect the views of the funding organizations.

1 INTRODUCTION

Significant advances in computing capabilities and the rising costs associated with physical experiments have contributed to an increase in both the use and complexity of numerical simulation. Often, these models are treated as an objective function to be optimized, such as in the design and control of complex engineering systems. The optimization is characterized by an inability to calculate derivatives and by the expense of obtaining a realization from the objective function. Due to the cost of simulation, it is essential that the optimization converges relatively quickly. A search of the magnitude required to guarantee global convergence is not feasible. But at the same time, these large engineering problems are often multi-modal and it is possible to get stuck in low quality solutions. We thus wish to take advantage of existing local optimization methods (i.e., algorithms which locate a function optimum nearby to a specified start location) for quick convergence, but use a statistical analysis of the entire function space to facilitate a global search and provide more robust solutions.

We argue for the utility of using predicted objective function output over unobserved input locations, through statistical emulation in the spirit of the analysis of computer experiments (e.g., Kennedy and O'Hagan, 2001; Santner, Williams, and Notz, 2004; Higdon, Kennedy, Cavendish, Cafeo, and Ryne, 2004), to act as a guide for underlying local optimization. Our framework could thus be classified as an *oracle* optimization approach (see Kolda, Lewis, and Torczon, 2003 and references therein), where information from alternative search schemes is used to periodically guide a relatively inexpensive local optimization. In particular, we propose a hybrid algorithm, referred to as TGP-APPS, which uses prediction based on nonstationary treed Gaussian process (TGP) modeling to influence asynchronous parallel pattern search (APPS) through changes to the search pattern. Based upon the predicted improvement statistics (see e.g. Schonlau, Welch, and Jones, 1998) at a dense random set of input locations, candidate points are ranked using a novel recursive algorithm and a predetermined number of top-ranked points are added to the search pattern. Both APPS and the TGP-based generation of candidate locations result in discrete sets of inputs that are queued for evaluation, and the merging of these two search patterns provides a natural avenue for communication between components. This same property makes the methodology easily parallelizable and efficient to implement. We argue that, in many situations, the approach will offer a robust and effective alternative to algorithms based only on either a global statistical search or a local pattern search. In addition, although we will refer specifically to APPS and TGP throughout and have experienced success with these methods, our parallel scheme encompasses a general approach to augmenting local search patterns with statistically generated location sets, and we emphasize that other researchers may find success using alternative methods for statistical emulation or for parallel search.

The methodological components underlying our approach, local optimization through asynchronous parallel pattern search and statistical emulation of the objective function with treed Gaussian processes, are described in Sections 2.1 and 2.2 respectively. The novel hybrid algorithm, combining APPS with TGP, is presented in Section 3. Details for the generation of ranked global search patterns based on statistical emulation are contained in Section 3.1. Section 3.2 discusses an initial design framework, including an informed sampling of the input space and sensitivity analysis. Section 3.3 outlines a framework for the asynchronous parallel implementation of our hybrid optimization algorithm, and the results are shown on two standard test problems (which are introduced below in the next section). In Section 4, we illustrate our methods on our motivating example involving calibration of a circuit device simulator. Finally, in Section 5, we investigate convergence and begin to consider how statistical information can be used to assess the quality of converged solutions.

1.1 Examples

For illustration of the methodology throughout this paper, we consider two common global optimization test functions, the Rosenbrock and Shubert problems. Both involve minimization of a continuous response $f(\mathbf{x})$ over a bounded region for inputs $\mathbf{x} = [x_1, x_2] \in \mathbb{R}^2$. Specifically, the two dimensional Rosenbrock function is defined as

$$f(\mathbf{x}) = 100(x_1^2 - x_2)^2 + (x_1 - 1)^2, \tag{1}$$

where herein $-1 \leq x_i \leq 5$ for $i = 1, 2$, and the Shubert function is defined as

$$f(\mathbf{x}) = \left(\sum_{j=1}^{5} j \cos((j+1)x_1 + j) \right) \left(\sum_{j=1}^{5} j \cos((j+1)x_2 + j) \right), \tag{2}$$

where $-10 \leq x_i \leq 10$ for $i = 1, 2$. The global solution of the Rosenbrock problem is $\mathbf{x}^\star = (1, 1)$ for $f(\mathbf{x}^\star) = 0$, and the Shubert problem has 18 global minima $\mathbf{x}^\star$ with $f(\mathbf{x}^\star) = -186.7309$ (problem descriptions from Hedar and Fukushima, 2006).

The response surfaces are shown in the background of Figure 1. Some particular difficulties for optimization algorithms are emphasized in these problems. The plotted log-response surface for the Rosenbrock function shows a steep valley with a gradually sloping floor. The solution lies at the end of this long valley, and the combination of steep gradients up the valley walls and gradual gradients along the valley floor will typically cause problems for local search methods such as gradient descent and pattern search. As seen on the right hand side of Figure 1, the Shubert problem is characterized by rapid oscillations of the response surface. There are multiple minima and each is located adjacent to dual local maxima, thus presenting a challenging problem for global optimization methods that will tend to over explore the input space or miss a minimum hidden amongst the maxima.

Each of these problems illustrate different aspects of our algorithm. The Rosenbrock problem is specifically designed to cause local pattern search methods to break down. The motivation for considering this problem is to show the potential for TGP-APPS to overcome difficulties in the underlying pattern search, and to significantly decrease computation time in certain situations. Conversely, the Shubert problem is relatively easily solved through standard pattern search methods, while the presence of multiple global minima would cause many algorithms based solely on statistical prediction to over-explore the input space and lead to higher than necessary computation costs. Indeed, the APPS optimization does converge, on average, in about half the iterations used by TGP-APPS. However, this is a small increase in computation compared to that which would be required by many fully global optimization routines (such as genetic algorithms or simulated annealing). And the results for APPS-TGP may be considered more robust; the global scope of the TGP search protects against premature convergence to a local optimum. The benefit of this additional robustness is clearly illustrated in the real-world application of Section 4.

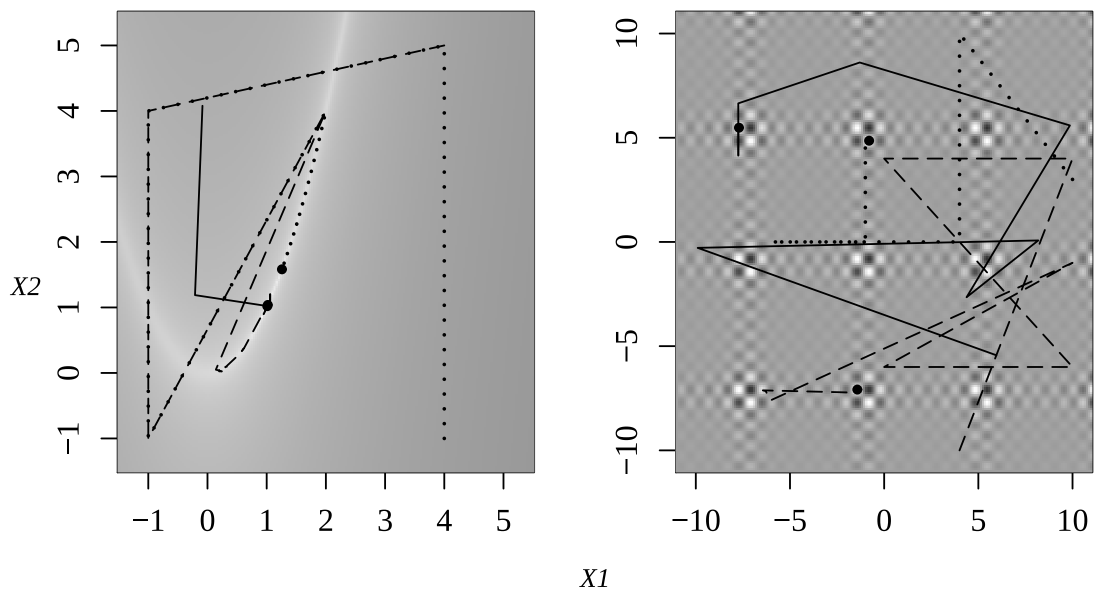

Figure 1: Rosenbrock (left) and Shubert (right) test problems. The response surface images (log response for Rosenbrock) are rising from white to black, and trace paths for the *best point* during optimization are shown as a dotted line for APPS, a dashed line for APPS-TGP, and a solid line for APPS-TGP following an initialization through Latin hypercube sampling.

2 ELEMENTS OF THE METHODOLOGY

The optimization algorithm is based on point locations suggested either by asynchronous parallel pattern search, or through a statistical analysis of the objective function based on treed Gaussian processes. These two methodological elements will be outlined in Sections 2.1 and 2.2, respectively. As is the case throughout, all algorithms and examples have minimization as the unstated goal.

2.1 Asynchronous parallel pattern search

Pattern search is included in a class of derivative-free optimization methods primarily developed to address problems in which the derivative of the objective function is unavailable and approximations are unreliable (Wright, 1996). The optimization uses a predetermined pattern of points to sample a given function domain, and is considered a direct search algorithm (Kolda et al., 2003) with no attempt made to explicitly evaluate or estimate local derivatives. This type of optimizer is considered to be more robust than derivative-based approaches for difficult optimization problems with nonsmooth, discontinuous, or undefined points.

The APPS algorithm is much more complicated than simple pattern search, requiring careful bookkeeping, and we thus only provide a brief outline of the basic steps in this paper, referring the more interested reader to Kolda (2005) and Gray and Kolda (2006). At each iteration k of APPS, three basic steps are executed:

1. generate a set of trial points Q_k around the current *best point* $\mathbf{x}_k^{best}$ (defined below),
2. send trial points Q_k to the compute cluster, and obtain a set of function evaluations R_k,
3. update the best point, $\mathbf{x}_{k+1}^{best}$.

Convergence to locally optimal points is ensured using a *sufficient decrease criterion* for accepting new best points. An incoming trial point $\mathbf{x}'$ is considered to be a new *best point* if $f(\mathbf{x}')-f(\mathbf{x}_k^{best}) < \delta$, for user defined $\delta > 0$. In the unconstrained case, using standard assumptions, it can be shown that

$$\lim_{k\to\infty}\inf \|\nabla f(\mathbf{x}_k^{best})\| \to 0.$$

Similar optimality results can be shown for linearly constrained optimization in terms of projections onto local tangent cones (Kolda, Lewis, and Torczon, 2006).

Trial points are generated using a positive spanning set of search directions $\{d_1, \ldots, d_L\}$ and have the form $Q_k = \{\mathbf{x}_k^{best} + \Delta_{kl} d_l \mid 1 \leq l \leq L\}$, for positive step sizes Δ_{kl}. The step sizes and search directions are chosen as described in Gray and Kolda (2006). After a successful iteration (one in which a new best point has been found), the step size is either left unchanged or increased. However, if the iteration was unsuccessful, the step size is reduced. A defining difference between simple pattern search and APPS is that, for APPS, directions are processed independently and each direction may have its own step size. Because of the asynchronous environment, the members of Q_k will generally not all be returned in R_{k+1} but instead will be spread throughout the next several R's. Thus APPS needs to be able to deal with the fact that the function evaluations may be returned in a different order than requested, and that the generation of the next iteration of trial points Q_{k+1} may need to be created before all of the results of the previous iteration are available.

This algorithm has been implemented in an open source software package called `APPSPACK` and has been successfully applied to problems in microfluidics, biology, groundwater, thermal design, and forging; see Gray and Kolda (2006) and references therein. The latest software is publicly available at `http://software.sandia.gov/appspack/`. There are many plausible competitors for APPS as a derivative-free optimization method; see, for example, Fowler et al. (2008), for a thorough comparison of a dozen such algorithms. However, it should be noted that it is the parallelization of the APPS search which makes it particularly amenable to a hybrid search scheme and that there are relatively few available parallel methods. Our software development is exploring the use of alternative parallel search components, but none of these are as easily available or as widely distributed as APPS.

2.2 Treed Gaussian process emulation

A Bayesian approach was brought to the emulation of computer code in the paper by Currin, Mitchell, Morris, and Ylvisaker (1991) which focuses on the commonly used Gaussian process (GP) model. As well, the book by Santner et al. (2003) follows a mainly Bayesian methodology and offers a detailed outline of its implementation through examples. The standard practice in the computer experiments literature is to model the output of the simulations as a realization of a stationary GP (Sacks, Welch, Mitchell, and Wynn, 1989; O'Hagan, Kennedy, and Oakley 1998; Fang, Li, and Sudjianto, 2006). In this setting, the unknown function is modeled as a stochastic process: the response is a random variable $f(\mathbf{x})$ dependent upon input vector $\mathbf{x}$. In model specification, the set of stochastic process priors indexed by the process parameters and their prior distributions represent our prior uncertainty regarding possible output surfaces. It is possible to model both deterministic and non-deterministic functions with these methods. The basic GP model is $f(\mathbf{x}) = \mu(\mathbf{x}) + w(\mathbf{x})$ where $\mu(\mathbf{x})$ is a simple mean function, such as a constant or a low-order polynomial, and $w(\mathbf{x})$ is a zero mean random process with covariance function $c(\mathbf{x}_i, \mathbf{x}_j)$. A typical approach would be to use a linear mean trend $\mu(\mathbf{x}) = \mathbf{x}\boldsymbol{\beta}$ and an anisotropic Gaussian correlation function,

$$c(\mathbf{x}_i, \mathbf{x}_j) = \exp\left[-\left(\sum_{k=1}^{d} \frac{(x_{ik} - x_{jk})^2}{\theta_k}\right)\right] \tag{3}$$

where d is the dimension of the input space and θ_k is the range parameter for each dimension.

Treed Gaussian process (TGP) models form a natural extension of this methodology and provide a more flexible nonstationary regression scheme (Gramacy and Lee, 2008). There is software available in the form of a `tgp` library for the statistical package R (see `http://www.cran.r-project.org/src/contrib/Descriptions/tgp.html`), which includes all of the statistical methods described in this paper. TGP models work by partitioning the input space into disjoint regions, wherein an independent GP prior is assumed. Partitioning allows for the modeling of nonstationary behavior, and can ameliorate some of the computational demand of nonstationary modeling by fitting separate GPs to smaller data sets (the individual partitions). The partitioning is achieved in a fashion derived from the Bayesian Classification and Regression Tree work of Chipman, George, and McCulloch (1998 & 2002). Our implementation uses reversible jump Markov chain Monte Carlo (Green, 1995) with tree proposal operations (prune, grow, swap, change, and rotate) to simultaneously fit the tree and the parameters of the individual GP models. In this way, all parts of the model can be learned automatically from the data, and Bayesian model averaging through reversible jump allows for explicit estimation of predictive uncertainty. The prior on the tree space is a process prior specifying that each leaf node splits with probability $a(1+q)^{-b}$, where q is the depth of the node and a and b are parameters chosen to give an appropriate size and spread to the distribution of trees. We use hierarchical priors for the GP parameters within each of the final leaf nodes ν. For each region ν, the hierarchical GP model is

$$\begin{aligned}
\mathbf{Z}_\nu|\boldsymbol{\beta}_\nu, \sigma_\nu^2, \mathbf{K}_\nu &\sim N_{n_\nu}(\mathbf{F}_\nu\boldsymbol{\beta}_\nu, \sigma_\nu^2\mathbf{K}_\nu), & \boldsymbol{\beta}_0 &\sim N_{d+1}(\boldsymbol{\mu}, \mathbf{B}) \\
\boldsymbol{\beta}_\nu|\sigma_\nu^2, \tau_\nu^2, \mathbf{W}, \boldsymbol{\beta}_0 &\sim N_{d+1}(\boldsymbol{\beta}_0, \sigma_\nu^2\tau_\nu^2\mathbf{W}) & \tau_\nu^2 &\sim IG(\alpha_\tau/2, q_\tau/2), \\
\sigma_\nu^2 &\sim IG(\alpha_\sigma/2, q_\sigma/2), & \mathbf{W}^{-1} &\sim W((\rho\mathbf{V})^{-1}, \rho),
\end{aligned}$$

with $\mathbf{F}_\nu = (\mathbf{1}, \mathbf{X}_\nu)$, and $\mathbf{W}$ is a $(d+1)\times(d+1)$ matrix. The N, IG, and W are the (Multivariate) Normal, Inverse-Gamma, and Wishart distributions, respectively. Hyperparameters $\boldsymbol{\mu}$, $\mathbf{B}$, $\mathbf{V}$, ρ, α_σ, q_σ, α_τ, q_τ are treated as known, and we use the default values from the `tgp` package. The coefficients $\boldsymbol{\beta}_\nu$ are modeled hierarchically with a common unknown mean $\boldsymbol{\beta}_0$ and region-specific variance $\sigma_\nu^2\tau_\nu^2$. There is no explicit mechanism in this model to ensure that the process is continuous across the partitions. However, the model can capture smoothness through model averaging, as predictions are integrated over the tree space, so when the true function is smooth, the predictions will be as well.

When the data actually indicate a non-smooth process, the treed GP retains the flexibility necessary to model discontinuities. One further advantage of TGP over a standard GP is in computational efficiency – fitting a Bayesian GP requires repeated inversion of an $n \times n$ matrix (where n is the sample size), requiring $O(n^3)$ computing time. By partitioning the space, each region contains a smaller subsample, and so the computational effort is significantly reduced. Further details of implementation and properties for TGP are available in Gramacy and Lee (2008). We note that other emulators, such as neural networks, could be considered, however we prefer the GP/TGP family because of the ability to ensure a degree of smoothness in the fitted function, the ability to model nonstationarity, and the robustness in the fitting of the model.

The TGP model involves a complex prior specification in order to promote mixing over alternative tree structures. However, the prior parametrization provided as a default for the `tgp` software is designed to work *out-of-the-box* in a wide variety of applications. In all of the examples of this paper, including the circuit application of Section 4, after both input and response have been scaled to have mean of zero and a variance of one, process and tree parameters were assigned the default priors from the `tgp` software (Gramacy, 2007). Here each partition has a GP of the form (3) but with an additional nugget parameter in the correlation structure. Each range parameter θ_k has prior $\pi(\theta_k) = \text{gamma}(1, 20)/2 + \text{gamma}(10, 10)/2$, where $\text{gamma}(a, b)$ has expectation a/b. The covariance for $(\mathbf{x}_i, \mathbf{x}_j)$ within the same tree partition is then $\sigma^2(c(\mathbf{x}_i, \mathbf{x}_j) + \gamma)$, with σ^2 the GP variance and γ the nugget parameter. The nugget term requires the only non-default parametrization, due to the fact that we are modeling deterministic objective functions that do not involve random noise. Although we do not completely remove accommodation of random error from the model, γ is forced to be small through the prior specification $\pi(\gamma) = \text{gamma}(1, 100)$ (as opposed to the default $\text{gamma}(1, 1)$ for noisy response surfaces). The presence of a small nugget allows for smoothing of the predicted response surface to avoid the potential instability that is inherent in point-by-point interpolation. This smoothing is made possible through the hybridization; since the local optimization relies upon pattern search rather than the TGP predicted response, it is more important for the statistical modeling to be globally appropriate than for it to be a perfect interpolator. We note that the nugget also allows for the possibility of numerical instability in complex simulators, and it improves numerical stability of the covariance matrix inversions required during MCMC.

The evaluated iterates of the optimization, in addition to any initial sampling as outlined in Section 3.2, provide the data for which the TGP model is to be fit. Thus the statistical model will be able to learn throughout the algorithm, leading to improved prediction as the optimization proceeds. Mean posterior response surface estimates for the Rosenbrock and Shubert problems are shown in Figure 2 for TGP fit to an initial sample of 20 function evaluations, as well as to the entire set of evaluated iterates at the time of convergence for each test problem. This illustrates the considerable amount of information gained during optimization.

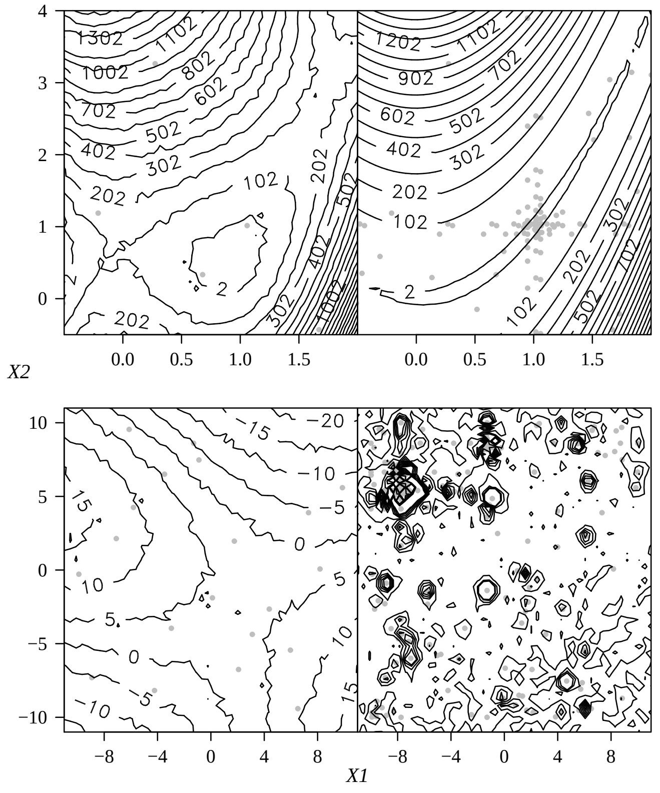

Figure 2: Posterior mean predictive response surfaces for Rosenbrock (top) and Shubert (bottom) problems. The left hand column corresponds to fit conditional on the initial sample of 20 points, and the right hand column corresponds to TGP fit to all of the iterations at the time of convergence (128 for Rosenbrock, 260 iterations for Shubert). In each case, the evaluated iterates are plotted in grey.

3 HYBRID STATISTICAL OPTIMIZATION

We now present a framework for having global prediction, through the TGP emulator, guide local pattern search optimization. Statistical methods have previously been employed in the optimization of expensive black-box functions, usually in the estimation of function response through interpolation. This estimated surface is then used as a surrogate model to be referenced during the optimization. Generally, in these *surrogate* or *response surface* based schemes, optimization methods are applied to the less expensive surrogate with periodic corrections from the expensive simulation to ensure convergence to a local optimum of the actual simulator (see e.g., Booker, Dennis, Frank, Serafini, Torczon, and Trosset, 1999; Alexandrov, Dennis, Lewis, and Torczon, 1998). The Expected Global Optimizer (EGO) algorithm developed by Jones, Schonlau, and Welch (1998) instead uses the surrogate model to provide input locations to be evaluated by the expensive simulator. At each iteration, a GP is fit to the set of function evaluations and a new location for simulation is chosen based upon this GP estimate. The method is designed to search the input space and converge towards the global optimum. This algorithm characterizes a general class of global algorithms that use response surface estimates to determine the input search. Similar algorithms based around radial basis function approximations have appeared recently in the literature (Regis and Shoemaker, 2007). The approach of Jones et al. is distinguished by the use of not only the response surface, but also the estimation error across this surface, in choosing new input locations. This is achieved through improvement statistics, and these feature prominently in our methods below.

While the optimization approach presented here has similarities to these response surface global algorithms, the underlying motivation and framework are completely distinct. Moreover, existing response surface methodologies rely either on a single point estimate of the objective surface or, in the case of the EGO and related algorithms, point estimates of the parameters governing the probability distribution around the response surface. Conversely, our analysis is fully Bayesian and will be fit using MCMC, providing a sample posterior predictive distribution for the response at any desired location in the input space. Full posterior sampling is essential to our algorithm for the ranking of a discrete input set, and since TGP modeling is not the sole source for new search information, the computational expense of MCMC prediction is acceptable. We argue that the cost associated with re-starts for a local optimization algorithm (a standard approach for checking robustness) will quickly surpass that of our more coherent approach in all but the fastest and smallest optimization problems. In combining APPS with TGP, our goal is to provide global scope to an inherently local search algorithm. The solutions obtained through the hybrid algorithm will be more robust than those obtained through any isolated pattern search. But, in addition, we have observed that the hybrid algorithm can lead to more efficient optimization of difficult problems.

This section will outline the main pieces of our optimization framework. We describe in Section 3.1 an algorithm to suggest additional search locations based upon a statistical analysis of the objective function, in Section 3.2 a framework for initial sampling of the input space and sensitivity analysis, and in Section 3.3 an outline of a general framework for efficient parallel implementation of such hybrid algorithms. Finally, Section 3.4 contains the optimization results for our example problems.

3.1 Statistically generated search patterns

We focus on the posterior distribution of improvement statistics, obtained through MCMC sampling conditional on a TGP model fit to evaluated iterates, in building a location set to augment the local search pattern. Improvement is defined here, for a single location, as $I(\mathbf{x}) = \max\{f_{min} - f(\mathbf{x}), 0\}$, where f_{min} is the minimum evaluated response in the search (f_{min} may be less than the response at the present *best point*, as defined above in Section 2.1 for purposes of the local pattern search, due to the *sufficient decrease* condition). In particular, we will use the posterior expectation of improvement statistics as a criterion for selecting input locations to be sent for evaluation. Note that the improvement is always non-negative, as points which do not turn out to be new minimum points still provide valuable information about the output surface. Thus, in the expectation, candidate locations will be rewarded for high response uncertainty (indicating a poorly explored region of the input space, such that the response could easily be lower than f_{min}) as well as for low mean predicted response.

Schonlau et al. (1998) provide an extensive discussion of improvement statistics, and also propose some variations on the standard improvement which will be useful in generating location sets. The exponentiated $I^g(\mathbf{x}) = (\max\{(f_{min} - f(\mathbf{x})), 0\})^g$, where g is a non-negative integer, is a more general improvement statistic. Increasing g increases the global scope of the criteria by rewarding in the expectation extra variability at $\mathbf{x}$. For example, $g = 0$ leads to $\mathbb{E}[I^0(\mathbf{x})] = \Pr(I(\mathbf{x}) > 0)$ (assuming the convention $0^0 = 0$), $g = 1$ yields the standard statistic, and $g = 2$ explicitly rewards the improvement variance since $\mathbb{E}[I^2(\mathbf{x})] = \text{var}[I(\mathbf{x})] + \mathbb{E}[I(\mathbf{x})]^2$. We have experienced some success with a g that varies throughout the optimization, decreasing as the process converges. In all of our examples, the algorithm begins with $g = 2$, and this changes to $g = 1$ once the maximum APPS step size ($\max\{\Delta_{k1}, \ldots, \Delta_{kL}\}$ from Section 2.1) drops below 0.05. However, we have found the algorithm to be robust to this choice, and a fixed g has not been observed to hurt performance. Expected improvement surfaces $\mathbb{E}[I^g(\mathbf{x})]$ with $g = 1$ and $g = 2$, for both the Shubert and Rosenbrock problems, are illustrated in Figure 3 conditional on a TGP fit to the first 75 iterates of an optimization run. The surfaces show only a subtle difference in structure across the different g parametrization, however this can lead to substantially different search patterns based on the ranking algorithm which we now describe.

The TGP generated search pattern will consist of m locations that maximize (over a discrete candidate set) the expected multi-location improvement, $\mathbb{E}\left[I^g(\mathbf{x}_1, \ldots, \mathbf{x}_m)\right]$, where

$$I^g(\mathbf{x}_1, \ldots, \mathbf{x}_m) = (\max\{(f_{min} - f(\mathbf{x}_1)), \ldots, (f_{min} - f(\mathbf{x}_m)), 0\})^g \tag{4}$$

(Schonlau et al., 1998). Finding the maximum expectation of (4) will, in most situations, be difficult and expensive. In particular, it is impossible to do so for the full posterior distribution of $I^g(\mathbf{x}_1, \ldots, \mathbf{x}_m)$, and would require conditioning on a single fit for the parameters of TGP. Our proposed solution is to discretize the d-dimensional input space onto a dense candidate set $\tilde{\mathbf{X}}$ of M locations. Although optimization over this set will not necessarily lead to the optimal solution in the underlying continuous input space, the use of APPS for local search means that such exact optimization is not required.

The discretization of decision space allows for a fast iterative solution to the optimization of $\mathbb{E}\left[I^g(\mathbf{x}_1, \ldots, \mathbf{x}_m)\right]$. This begins with evaluation of the simple improvement $I^g(\tilde{\mathbf{x}}_i)$ over $\tilde{\mathbf{x}}_i \in \tilde{\mathbf{X}}$ at each of T MCMC iterations (each corresponding to a single posterior realization of TGP parameters

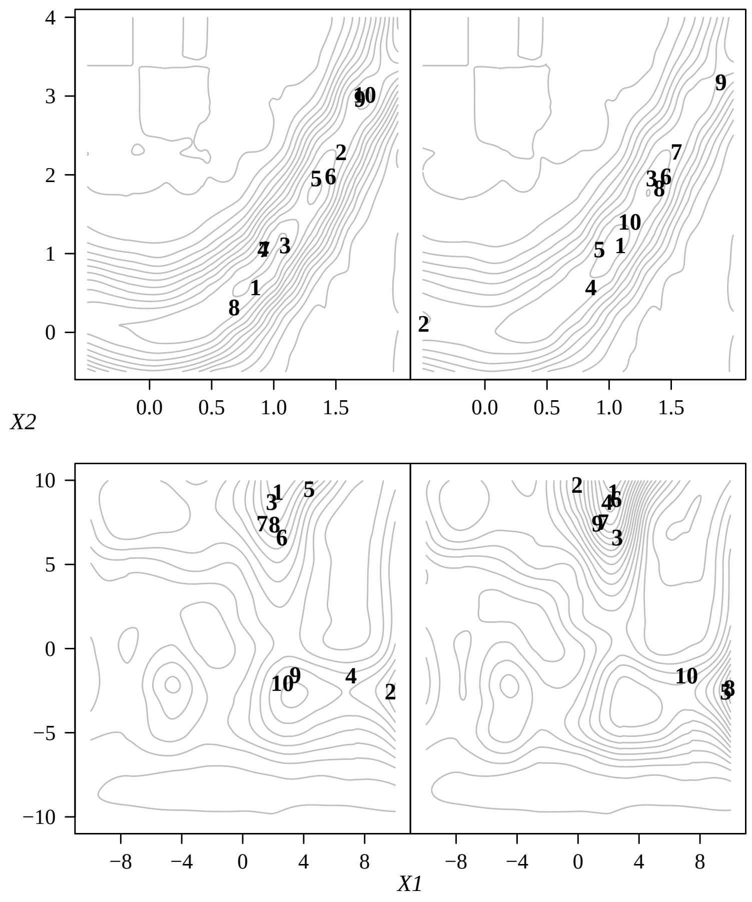

Figure 3: Expected improvement surfaces $\mathbb{E}[I^g(\mathbf{x})]$ and ranks of the top ten candidate locations, for the Rosenbrock (top) and Shubert (bottom) problems, conditional on TGP fit to 75 function evaluations. The left hand plots correspond to $g = 1$ and the right hand plots to $g = 2$.

and predicted response) to obtain the posterior sample

$$\mathcal{I} = \left\{ \begin{array}{ccc} I^g(\tilde{\mathbf{x}}_1)_1 & \dots & I^g(\tilde{\mathbf{x}}_M)_1 \\ & \vdots & \\ I^g(\tilde{\mathbf{x}}_1)_T & \dots & I^g(\tilde{\mathbf{x}}_M)_T \end{array} \right\}. \tag{5}$$

We then proceed iteratively to build an *ordered* search pattern of m locations: Designate $\mathbf{x}_1 = \text{argmax}_{\tilde{\mathbf{x}} \in \tilde{\mathbf{X}}} \mathbb{E}\left[I^g(\tilde{\mathbf{x}})\right]$, and for $j = 2, \ldots, m$, given that $\mathbf{x}_1, \ldots, \mathbf{x}_{j-1}$ are already included in the search pattern, the next member is

$$\begin{aligned}
\mathbf{x}_j &= \text{argmax}_{\tilde{\mathbf{x}}\in\tilde{\mathbf{X}}}\mathbb{E}\left[\max\{I^g(\mathbf{x}_1,\ldots,\mathbf{x}_{j-1}), I^g(\tilde{\mathbf{x}})\}\right] \\
&= \text{argmax}_{\tilde{\mathbf{x}}\in\tilde{\mathbf{X}}}\mathbb{E}[(\max\{(f_{min}-f(\mathbf{x}_1)),\ldots,(f_{min}-f(\mathbf{x}_{j-1})),(f_{min}-f(\tilde{\mathbf{x}})),0\})^g] \\
&= \text{argmax}_{\tilde{\mathbf{x}}\in\tilde{\mathbf{X}}}\mathbb{E}\left[I^g(\mathbf{x}_1,\ldots,\mathbf{x}_{j-1},\tilde{\mathbf{x}})\right].
\end{aligned}$$

Thus, after each j-th additional point is added to the set, we have the maximum expected j location improvement conditional on the first $j-1$ locations. This is not necessarily the unconditionally maximal expected j location improvement; instead, the point $\mathbf{x}_j$ is the location which will cause the greatest increase in expected improvement over the given $j-1$ location expected improvement.

Note that the above expectations are all taken with respect to the posterior sample $\mathcal{I}$, which acts as a discrete approximation to the true posterior distribution for improvement at locations within the candidate set. Hence, iterative selection of the point set is possible without any re-fitting of the TGP model. It follows that $\mathbf{x}_1 = \tilde{\mathbf{x}}_{i_1}$, the first location to be included in the search pattern, is such that the average of the i_1-th column of $\mathcal{I}$ is greater than every other column average. Conditional on the inclusion of x_{i_1} in the search pattern, a posterior sample of the two-location improvement statistics is calculated as

$$\mathcal{I}_2 = \left\{\begin{array}{ccc} I^g(\tilde{\mathbf{x}}_{i_1},\tilde{\mathbf{x}}_1)_1 & \ldots & I^g(\tilde{\mathbf{x}}_{i_1},\tilde{\mathbf{x}}_M)_1 \\ & \vdots & \\ I^g(\tilde{\mathbf{x}}_{i_1},\tilde{\mathbf{x}}_1)_T & \ldots & I^g(\tilde{\mathbf{x}}_{i_1},\tilde{\mathbf{x}}_M)_T \end{array}\right\}, \tag{6}$$

where the element in the t-th row and j-th column of this matrix is calculated as $\max\{I^g(\tilde{\mathbf{x}}_{i_1})_t, I^g(\tilde{\mathbf{x}}_j)_t\}$. The second location in the search pattern, $\mathbf{x}_2 = \tilde{\mathbf{x}}_{i_2}$, is then chosen such that the i_2-th column of $\mathcal{I}_2$ is the column with greatest mean. Similarly, $\mathcal{I}_l$, for $l = 3,\ldots,m$, has element (t,j) equal to $\max\{I^g(\tilde{\mathbf{x}}_{i_1},\ldots,\tilde{\mathbf{x}}_{i_{l-1}})_t,\ I^g(\tilde{\mathbf{x}}_j)_t\} = I^g(\tilde{\mathbf{x}}_{i_1},\ldots,\tilde{\mathbf{x}}_{i_{l-1}},\tilde{\mathbf{x}}_j)_t$ and the l-th location included in the search pattern corresponds to the column of this matrix with maximum average. Since the multi-location improvement is always at least as high as the improvement at any subset of those locations, the same points will not be chosen twice for inclusion.

An appealing byproduct of this technique is that the search pattern has been implicitly ordered, producing a ranked set of locations that will be placed in the queue for evaluation. The ten top-ranked points corresponding to this algorithm, based on a TGP fit to the first 75 iterations of Rosenbrock and Shubert problem optimizations, are shown in Figure 3. We note that the rankings are significantly different depending on whether $g = 1$ or $g = 2$.

For this method to be successful, the candidate set needs to be suitably dense over the input space. In the physics and engineering problems that motivate this paper, there is typically prior information from experimentalists or modelers on where the optimum would be found. This information can be expressed through a probability density $u(\mathbf{x})$ over the input space, which will be referred to throughout as the *uncertainty distribution*. In the case that the uncertainty distribution is bounded, as is standard, the candidate set can be drawn as a Latin hypercube sample (LHS, e.g. McKay, Beckman, and Conover, 1979) proportional to u. This produces a space filling design that is more concentrated in areas where the optimum is expected, and is thus an efficient way to populate the candidate set. There are many different LHS techniques available, and Stein (1987) discusses approaches to obtain LHS for variables that are either independent or dependent in u. In the event of strong prior information, techniques such as Latin Hyperrectangles can be used (Mease and Bingham, 2006). In practice, the prior beliefs regarding optimum inputs are often expressed independently for each variable in the form of bounds and, possibly, a point-value *best guess*. The sampling design is then easily formed by taking independent LHS in each dimension of the domain, either uniform over the variable bounds or proportional to a scaled and shifted Beta distribution with mode at the prior best guess.

We have also found it efficient to augment the large LHS of candidate locations $\tilde{\mathbf{X}}_{LHS}$ with a dense sample of locations $\tilde{\mathbf{X}}_B$ from a rectangular neighborhood around the present best point. The combined candidate set $\tilde{\mathbf{X}}$ has then been drawn proportional to a distribution based on prior beliefs about the optimum solution with an additional point mass placed on the region around the present best point. In the examples and applications presented in this paper, the candidate set is always an LHS of size 50 times the input dimension, taken with respect to a uniform distribution over the bounded input space, augmented by an additional 10% of the candidate locations taken from a smaller LHS bounded to within 5% of the domain range of the present best point.

3.2 Initial design and sensitivity analysis

A search of the input space is commonly used before optimization begins in order to tune algorithm parameters (such as error tolerance) and choose starting locations. The variety of search designs employed to this end includes simple random sampling, regular grids, and orthogonal arrays, among other approaches. If a statistician is to be involved in the optimization, the initial set of function evaluations is additionally desirable to inform statistical prediction. Seeding TGP through a space filling initial design (as opposed to using only points generated by APPS) ensures that the initial sample set is not concentrated in a local region of the input space. From this perspective, the initial search is a designed experiment over the input space. The literature on this subject is vast (see for example, Santner et al., 2003, Chap 5, and references therein) and specific application could depend on the modeling approach. In all of our work, the initial sample is drawn as an LHS proportional to the uncertainty distribution as described above in Section 3.1. An initial sample size of ten times the input dimension has been found to be successful in practice.

In addition to acting as a training set for the statistical emulator, an initial sample of function evaluations may be used as the basis for a global sensitivity analysis (SA), which resolves the sources of objective function output variability by apportioning elements of this variation to different sets of input variables (Saltelli, Ratto, Andres, Campolongo, Cariboni, Gatelli, Saisana, and Tarantola, 2008). In particular, the variability of the response is investigated with respect to variability of inputs as dictated by u, the *uncertainty distribution* (as in Section 3.1, this refers to a prior distribution on the location of the optimum). In large engineering problems there can be a huge number of input variables over which the objective is to be optimized, but only a small subset will be influential within the confines of their uncertainty distribution. Thus, SA is important for efficient optimization and it may be performed, at relatively little additional cost, on the basis of a statistical model fit to the initial sample. Two influential sensitivity indices, which will be useful in this setting, are the first-order sensitivity for the jth input variable, $S_j = \mathrm{var}_u\left(\mathbb{E}_u\left[f(\mathbf{x})|x_j\right]\right)/\mathrm{var}_u(f(\mathbf{x}))$, and the total sensitivity for input j, $T_j = \mathbb{E}_u\left[\mathrm{var}_u\left(f(\mathbf{x})|\mathbf{x}_{-j}\right)\right]/\mathrm{var}_u(f(\mathbf{x}))$. The first-order indices measure

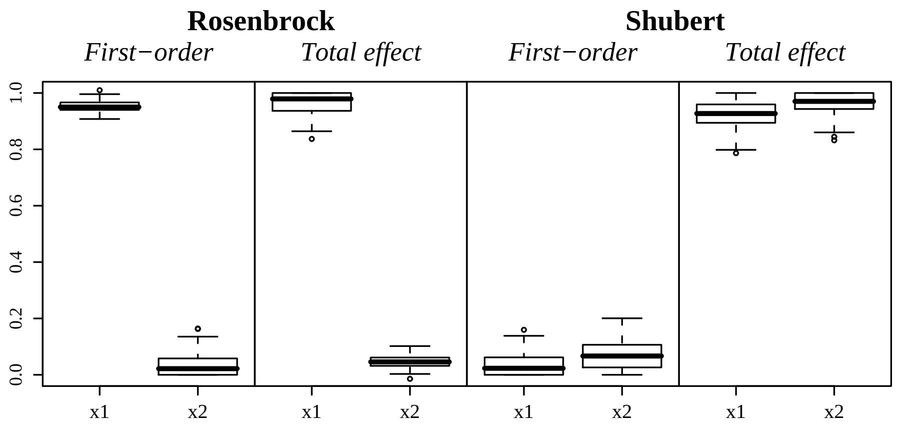

Figure 4: Sensitivity analysis of the test functions, with respect to independent uniform uncertainty distributions over each input range, from a TGP fit to an LHS of 20 initial locations, summarized by samples of the first order and total sensitivity indices.

the portion of variability that is due to variation in the main effects for each input variable, while the total effect indices measure the total portion of variability that is due to variation in each input. The difference between T_j and S_j provides a measure of the variability in $f(\mathbf{x})$ due to interaction between input j and the other input variables, and a large difference may lead the investigator to consider other sensitivity indices to determine where this interaction is most influential. Refer to Chapter 8 of Saltelli, Chan, and Scott (2000) and Saltelli et al. (2008) for descriptions of the complete analysis framework and practical guidelines on the use of sensitivity analysis.

Estimation of the integrals needed in calculation of the sensitivity indices usually requires a large number of function evaluations. Saltelli (2002) describes an efficient LHS based scheme for estimation of both first-order and total effect indices, but the required number of function evaluations will still be prohibitively large for applications involving an expensive simulation. However, using predicted response from the statistical emulator in place of true objective function response, it is possible to obtain sensitivity index estimates conditional on only the initial sample of function evaluations. Our approach is to use the Monte Carlo estimation procedure of Saltelli (2002) in conjunction with prediction from the TGP statistical emulator, conditional on an initial sample of function evaluations. At each iteration of the MCMC model fitting, response predictions are drawn over a set of input locations generated as prescribed in Saltelli's scheme, and estimates for the indices are calculated based upon this predicted response. The procedure was performed for the two example problems, and results are shown in Figure 4. From the large difference between first-order and total sensitivity indices for the Shubert function, it is clear that interaction between the two input variables is responsible for almost all of the variability in response. Conversely, the sample of Rosenbrock function total sensitivity indices is virtually identical to the sample of first-order indices. This is caused by a response variability that is dominated by change due to variation in x_1 (i.e., dominated by the steep valley walls of the response surface). If such a situation occurred in an expensive real-world optimization, one may wish to fix x_2 at a prior guess in order to avoid a lengthy optimization along the gradually sloping valley floor. We feel that this MCMC approach to SA is appealing due to the full posterior sample obtained for each sensitivity index, but one could also use maximum likelihood or empirical Bayes methodology to analytically calculate the indices of interest conditional on a statistical model fit to the initial sample, as in Morris, Kottas, Taddy, Furfaro, and Ganapol (2008) or Oakley and O'Hagan (2004).

3.3 Parallel computing environment

Our approach involves three distinct processes: LHS initialization, TGP model fit and point ranking via MCMC, and APPS local optimization. The hybridization used in this paper is loosely coupled, in that the three different components run independently of each other. This is beneficial from a software development perspective, since each component can be based on the original source code (in this case, the publicly available `APPSPACK` and `tgp` software). Our scheme is a combination of sequential and parallel hybridization; the algorithm begins with LHS sampling, followed by TGP and APPS in parallel. APPS and TGP run, for the most part, independently of each other. TGP relies only upon the growing cache of function evaluations, with the sole tasks during MCMC being model fit and the ranking of candidate locations. Similarly, ranked input sets provided by TGP are interpreted by APPS in an identical fashion to other trial points and are ignored after evaluation unless deemed better than the current best point. Thus, neither algorithm is aware that a concurrent algorithm is running in parallel. However, the algorithm does contain an integrative component in the sense that points submitted by TGP are given a higher priority and are hence placed at the front of the queue when available.

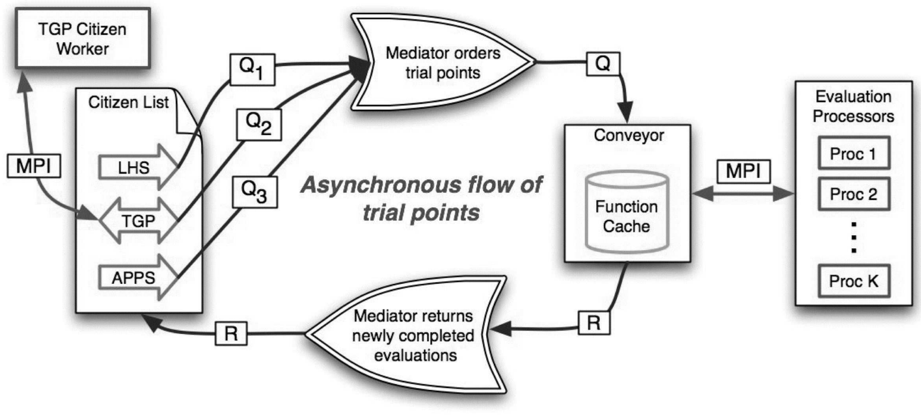

Figure 5: Schematic illustration of the hybrid parallel implementation for TGP-APPS (MPI is the *message passing interface*, a protocol for parallel computation.

We support hybrid optimization using a mediator/citizen/conveyor paradigm. Citizens are used to denote arbitrary optimization tools or solvers: in this case, LHS, TGP, and APPS. They communicate with a single mediator object, asynchronously exchanging unevaluated trial points for completed function evaluations. The mediator ensures that points are evaluated in a predefined order specified by the user, iteratively sending trial points from an ordered queue to free processors via a conveyor object. Three basic steps are performed iteratively: 1) exchange evaluated points for unevaluated points with citizens, 2) prioritize the list of requested evaluations, and 3) exchange unevaluated points for evaluated points. This process continues until either a maximum budget of evaluations has been reached or all citizens have indicated they are finished. The conveyor seeks to maximize the efficient use of available evaluation processors and to minimize processor idle time. The conveyor also maintains a function value cache that lexicographically stores a history of all completed function evaluations using a splay tree as described in Gray and Kolda (2006); this prevents a linear increase in look up time. Thus, prior to submitting a point to be evaluated, the cache is queried to ensure that the given point (or a very close location) has not previously been evaluated. If the point is not currently cached, a second query is performed to determine if an equivalent point is currently being evaluated. If the trial point is deemed to be completely new, it is then added to the evaluation queue. Equivalent points are just assigned the previously returned (or soon to be returned) objective value.

In the version of this algorithm utilized herein, the LHS citizen is given highest priority and hence has all points evaluated first. Because TGP is, in the presented applications, slower to submit points than APPS, it is given the second highest priority. Finally, the APPS citizen has free use of all evaluation processors not currently being used by either LHS or TGP. In this paradigm, TGP globalizes the search process while APPS is used to perform local optimization. The APPS local convergence occurs naturally due to the time spent during MCMC in which no points are being generated by TGP. Conveniently, due to a growing cache and thus more computationally intensive MCMC, the available window for local convergence increases during progression of the optimization. Although we have found this scheme to be quite successful, it may be necessary in other applications to allow APPS generated trial points to be given priority over TGP points when local convergence is desired (e.g., when approaching the computational budget or when the probability of improvement in a new area of the domain is sufficiently small).

Figure 5 illustrates the flow of trial points from the citizens, through the mediator, to the conveyor, and back to the citizens. The stream of trial-points is asynchronous in the sense that citizens can always submit points at each iteration. Here Q_1, Q_2, and Q_3 denote trial points submitted by citizens LHS, TGP, and APPS respectively, while Q is used to denote the ordered list of trial points given to the conveyor and R stores recently completed evaluations. Note that the TGP citizen worker stores a copy of the cache, since it lives on its own processor. In the illustrated algorithm, $K + 2$ processors are being used: one for the mediator, conveyor and APPS; one for the TGP citizen worker; and K evaluation processors. Present implementations of APPS cannot efficiently use more than $2d + 1$ evaluation processors, with d the dimension of the optimization problem, however extra available processors will always be useful to evaluate additional points from the TGP generated search pattern. Three processors are the minimum required for proper execution of the algorithm: one which serves as the mediator, one for APPS, and one for TGP.

3.4 Results for example problems

We have recorded the results from ten optimizations of each example function, using stand-alone APPS as well as the hybrid TGP-APPS, both with and without an initial LHS of 20 points. Each of these were run on seven of the eight available computation nodes on a Mac Pro with 2 quad core 3.2 GHz processors. Parametrization of the algorithm follows the directions detailed in Sections 2.2 and 3.1, and each TGP generated search pattern consists of 20 locations. Except when initial sampling is used to determine a starting location, the initial guess for each problem is $\mathbf{x} = (4, 4)$. In order to replicate the situation of a relatively expensive objective function, random wait times of between five and ten seconds were added to each function evaluation.

Table 1 contains average solutions and number of evaluations, and Figure 1 shows a selection of traces of $\boldsymbol{x}^{best}$ plotted over the response surfaces. For the Rosenbrock problem, we see that APPS required a vast increase in computational expense over TGP-APPS methods and was unable to locate the global minimum. Thus, in a problem designed to be difficult for local optimization methods, the hybrid algorithm offers a clear advantage by allowing the search to jump to optimal areas of the input domain. Conversely, the Shubert function is designed to be especially problematic for global optimization algorithms. Indeed, we observe that TGP-APPS required more iterations to locate a minimum than APPS alone. However, the increase in the number of evaluations is much less than one would observe for a truly global optimization algorithm such as simulated annealing or genetic search. Note that the higher average Shubert solution for APPS is due only to one particularly poor optimization wherein the pattern search converged on a local minimum of -48.51 after 28 iterations. The potential for such non-optimal convergence highlights the advantage of extra robustness and global scope provided by the hybrid TGP-APPS, with the cost being a doubling of the required function evaluations.

Figure 6 shows an objective function response trace during TGP-APPS (with initial LHS) optimization of the Rosenbrock and Shubert problems. The Rosenbrock trace plot is particularly informative, highlighting a property of TGP-APPS that we have repeatedly observed in practice. In the early part of the optimization, where there is much room for improvement over the present best point, the emulator chosen locations correspond to decent response values and are commonly found to be the next best point. As the optimization approaches convergence, the improvement values are driven entirely by predictive uncertainty, leading to a wide search of the input space and almost no probability that the next best point will have been proposed by the emulator. For

Table 1: Rosenbrock and Shubert problem results. For each optimization algorithm, the average and standard deviation (in brackets) for the number of function evaluations required and objective response at converged solution from a sample of ten runs.

Method	Problem	Evaluations	Solution
APPS	Rosenbrock	13918.3 (331.7)	0.0664 (0)
APPS-TGP (no LHS)	Rosenbrock	965.2 (845.7)	0.0213 (0.0162)
APPS-TGP (with LHS)	Rosenbrock	526.9 (869.3)	0.0195 (0.0227)
APPS	Shubert	81.2 (18.9)	-172.91 (43.7)
APPS-TGP (no LHS)	Shubert	206.0 (46.75)	-186.73 (0)
APPS-TGP (with LHS)	Shubert	180.7 (47.03)	-186.73 (0)

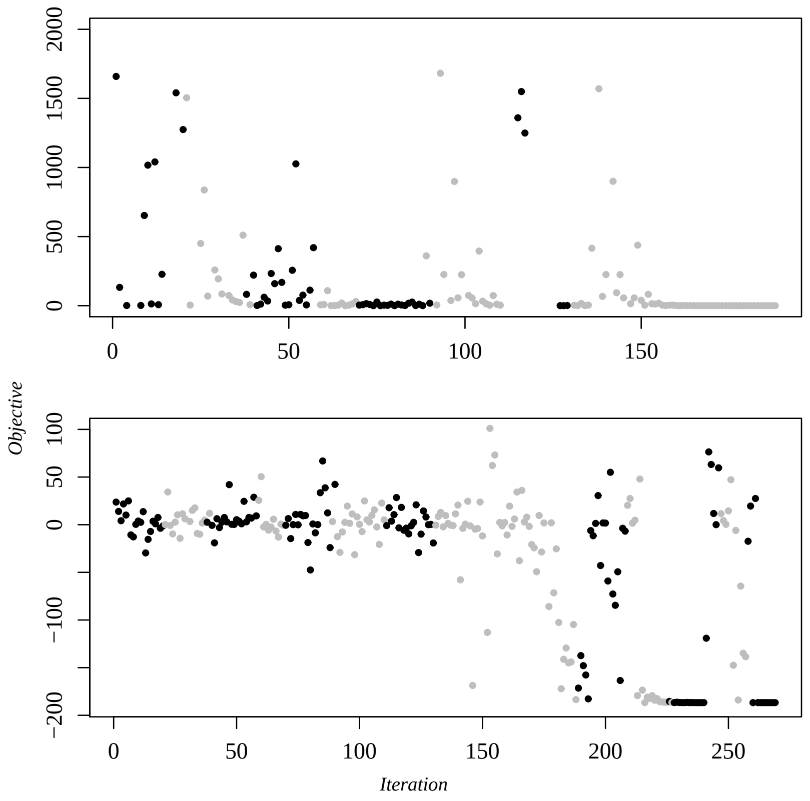

Figure 6: Objective function response trace during TGP-APPS optimization of the Rosenbrock (top) and Shubert (bottom) problems. Black points were generated by the TGP ranking algorithm, and grey points are generated by the APPS search component.

example, in the Rosenbrock optimization, most of iterations 111 to 130 (corresponding to points generated by the TGP ranking algorithm) led to objective response greater than 2000 (and do not appear on the plot). When this happens, the efficient local optimization of APPS will drive the optimization, leading to a quick and precise local convergence. In this way, each of APPS and TGP contribute their strengths to the project. Indeed, it may be desirable in future applications to devise a scheme whereby the priority for evaluation of TGP generated locations is downgraded once the posterior probability of non-zero improvement reaches some minimal level.

4 OPTIMIZATION OF A TRANSISTOR SIMULATOR

We now discuss the optimization problem of calibrating a radiation-aware simulation model for an electrical device component of a circuit. In this case, the goal of the optimization is to find appropriate simulator parameter values such that the resulting simulator output is as close as possible (under the distance function defined in (7), below) to results from real-world experiments involving the electrical devices of interest. The model input is a radiation pulse expressed as dose rate over time. The corresponding output is a curve of current value over time which reflects the response of the electrical device. The electrical devices, both bipolar junction transistors (bjts), are the *bft92a* and the *bfs17a*. Bjts are widely used in the semiconductor industry to amplify electrical current (refer to Sedra and Smith (1997) or Cogdell (1999) for more information), and the two in this study are particularly common. All bjts share a basic underlying model, and thus the same computer simulator can be used to approximate their behavior, with only changes to the tuning parameters required. The real-world data, to which simulated current response is compared, consists of six observations taken at a variety of testing facilities. In each experiment, the devices of interest are exposed to a unique radiation photo-current pulse and the resulting current behavior is recorded. Additional details about the experiments carried out, the experimental process, and the facilities used can be found in Gray et al. (2007). It should also be noted that selecting an appropriate subset of the real-world data was a problem unto itself and is described in Lee, Taddy, and Gray (2008).

The particular simulator of interest is a Xyce implementation of the Tor Fjeldy photo-current

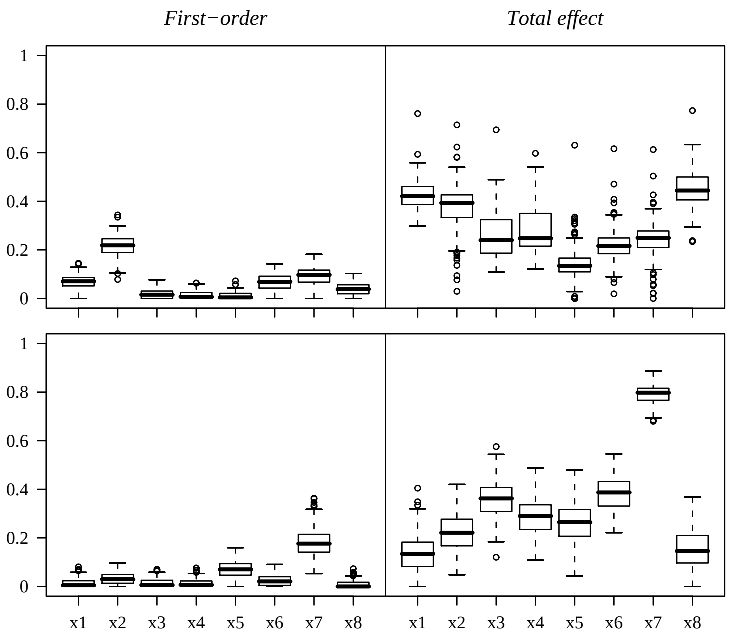

Figure 7: Sensitivity analysis for *bft92a* (top) and *bfs17a* (bottom) devices, based on a TGP fit to an LHS of 160 initial locations, summarized by first order sensitivity indices (left) and total sensitivity indices (right).

model for the bjt. Xyce is an electrical circuit simulator developed at Sandia National Laboratories (Keiter, 2004), and the Tor Fjeldy photo-current model is described in detail in Fjeldy, Ytterdal, and Shur (1997). There are 38 user-defined tuning parameters which determine simulator output for a given radiation pulse input. The objective function for optimization is the following measure of distance between simulator output and experimental observation of current paths in time for a set of specific radiation pulse inputs:

$$f(\mathbf{x}) = \sum_{i=1}^{N} \frac{1}{T_i} \sum_{t=1}^{T_i} \left[(S_i(t; \mathbf{x}) - E_i(t))^2 \right]. \tag{7}$$

Here, $N = 6$ is the number of experiments (each corresponding to a unique radiation pulse), T_i is the total number of time observations for experiment i, $E_i(t)$ is the amount of electrical current observed during experiment i at time t, and $S_i(t; \mathbf{x})$ is the amount of electrical current at time t as computed by the simulator with tuning parameters $\mathbf{x}$ and radiation pulse corresponding to experiment i.

Through discussion with experimentalists and researchers familiar with the simulator, 30 of the tuning parameters were fixed in advance to values either well known in the semiconductor industry or determined through analysis of the device construction. The semiconductor engineers also provided bounds for the remaining eight parameters, our objective function input $\mathbf{x}$. This parameter set includes those that are believed to have both a large overall effect on the output of the model and a high level of uncertainty with respect to their ideal values. The most uncertain parameters in the radiation-aware model are those that directly affect the amount of radiation collected, and these eight parameters are all part of the device doping profile. The doping process introduces impurities to change the electrical properties of the device and the amount of doping will affect the amount of photo-current that is collected by the device. Four of the parameters (x_1, x_2, x_3, x_4) describe the lightly doped collector while the other four (x_5, x_6, x_7, x_8) describe the heavily doped collector. Figure 7 shows the results of an MCMC sensitivity analysis, as described in Section 3.2, based on a TGP model fit to an initial LHS of 160 input locations and with respect to a uniform uncertainty distribution over the bounded parameter space. As the mean total effect indices are all above 0.1, it was decided that further reduction of the problem dimension was impossible. We note that, as indicated by the difference between first-order and total sensitivity indices, some variables are only influential in interaction with the other inputs.

The objective of (7) was optimized using both APPS and the hybrid algorithm TGP-APPS. The problems were initiated with the same starting values, the best guess provided by the semiconductor engineer, and were run on a cluster using 8 compute nodes with dual 3.06 GHz Intel Xenon processors with 2 GB of RAM. In the case of the hybrid algorithm, initial Latin hypercube samples of 160 points were used to inform the TGP. All LHS designs are taken with respect to independent uniform distributions over the bounded input space. The wall clock time and the number of objective function evaluations corresponding to each device and each optimization algorithm are shown in Table 3. Figure 8 shows simulated current response curves corresponding to each solution and to the initial guess for tuning parameter values, as well as the data, for a single radiation pulse input to each device. The initial guess represents the best values known to the experimentalists before the optimization was done. Results for the other radiation pulse input values exhibit similar properties.

In the case of *bft92a*, the solutions produced by the two optimization algorithms are practically

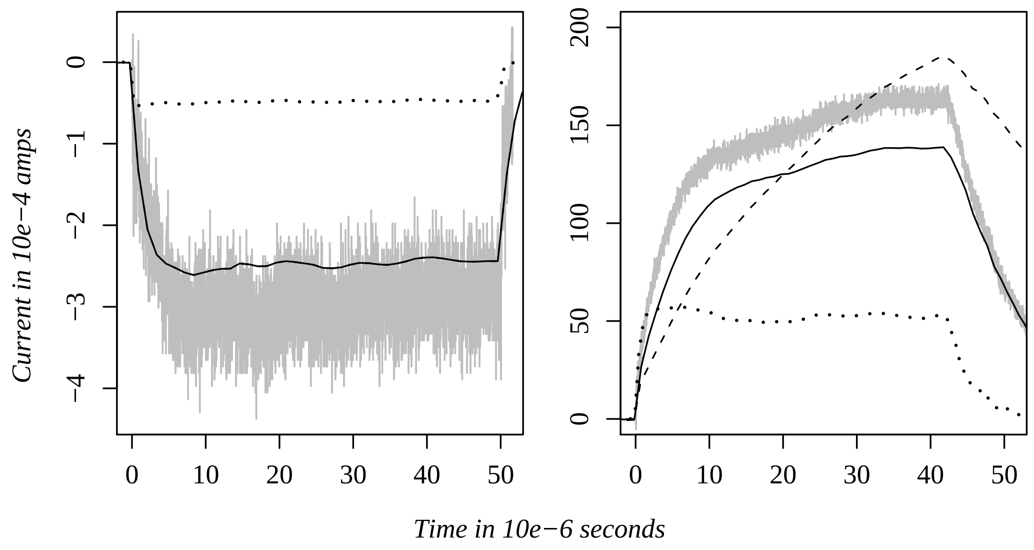

Figure 8: Simulated response current curves, corresponding to different tuning parameter values, for the *bft92a* (left) and *bfs17a* (right) devices. The solid line shows response for parameters found using TGP-APPS, the dashed line for parameters found through APPS alone, and the dotted line for the initial parameter vector guess. The experimental current response curves for the radiation impulse used in these simulations is shown in grey.

Table 2: Initial guesses and final solutions for *bft92a* and *bfs17a* simulator parameters.

bft92a	Initial	APPS	TGP-APPS	*bfs17a*	Initial	APPS	TGP-APPS
x_1	5e-3	3.55e-3	3.55e-3	x_1	5e-3	3.55e-3	3.55e-3
x_2	1.4e-3	1.08e-3	1.30e-3	x_2	1.4e-3	1.30e-3	1.30e-3
x_3	1e-8	1.00e-9	1.00e-9	x_3	1e-6	1.00e-6	1.00e-9
x_4	2e-8	6.15e-8	6.40e-8	x_4	2e-6	1.00e-5	1.00e-5
x_5	4e-3	3.55e-3	2.63e-3	x_5	4e-3	3.55e-3	3.55e-3
x_6	1.6e-3	1.30e-3	1.30e-3	x_6	1.6e-3	1.30e-3	1.20e-3
x_7	1e-9	1.61e-7	2.17e-7	x_7	1e-7	1.00e-5	1.19e-6
x_8	2e-9	1.00e-5	1.00e-5	x_8	2e-7	2.00e-7	1.00e-5

Table 3: For each bjt device and each optimization algorithm, the number of objective function evaluations, total wall clock time required to find a solution, and the optimized objective response.

Method	Device	Evaluations	Time	Objective
APPS	*bft92a*	6823	341257 sec $\approx$ 95 hrs	2.01644e-2
APPS-TGP	*bft92a*	962	49744 sec $\approx$ 14 hrs	2.01629e-2
APPS	*bfs17a*	811	37038 sec $\approx$ 10 hrs	3.22625e-2
APPS-TGP	*bfs17a*	1389	65122 sec $\approx$ 18 hrs	2.73194e-2

indistinguishable. However, the APPS solution was only obtained after a huge additional computational expense, illustrating the ability of the hybrid algorithm to move the search pattern quickly into decent areas of the input space. We note that a similarly low computational time (56815 seconds $\approx$ 16 hours) was required to obtain an equivalent solution through TGP-APPS without the initial sampling (i.e., starting from the same parameter vector as for APPS).

For the *bfs17a*, the difference in the resulting response curves is striking and illustrates the desirable robustness of our hybrid algorithm. The response curve created using the parameter values obtained by APPS alone differs significantly from the data in overall shape. In contrast, the curve resulting from the parameters found by TGP-APPS is a better match to the experimental data. These results suggest that the APPS algorithm was unable to overcome a weak local minimum while the inclusion of TGP allowed for a more comprehensive search of the design space. Overall, the APPS-TGP required more computational resources to find a solution. However, due to the poor quality of the APPS solution, the optimization would have needed to be re-run using different starting points until a more optimal solution were found, leading to computational cost quickly surpassing that of TGP-APPS. Thus, the extra computational cost of TGP-APPS is well justified by the improvement in fit and the ability to find a robust solution with just one starting point.

Table 2 shows the converged optimal parametrization for each device through the use of each of APPS and TGP-APPS with an LHS initial sample. As illustrated in Figure 8, a perfect solution was unobtainable. The lack of fit is partially attributed to a disconnect between the actual and simulated radiation pulses – a pre-specified pulse appears to tend to be larger than expected in the lab, leading to a muted simulation response curve for equivalent doses. This situation may also have led to incorrect bounds on the potential input domain. However, both the modelers and experimentalists were pleased with the improvement in fit provided by TGP-APPS. In fact, for this real-world problem, the solutions presented in this paper are the best known to date. A complete statistical calibration of this simulator would require the modeling of a bias term, as in the work of Kennedy and O'Hagan (2001); however, in the context of optimum control, we are confident that these results provide a robust solution with respect to minimization of the provided objective function. Indeed, the robustness of TGP-APPS allows the modelers to be confident that these solutions are practically *as close as possible* to the true current response curves. This discovery has convinced Sandia that the issues behind these inadequacies in the simulator model will merit future study.

5 CONCLUSION

We have described a novel algorithm for statistically guided pattern search. Along with the general optimization methodology described in Sections 3.2 and 3.3, this work outlines a powerful framework within which the strengths of both statistical inference and pattern search are utilized. The general hybridization scheme, as well as the algorithm for statistically ranking candidate locations, do not require the use of APPS or TGP specifically and could be implemented in conjunction with alternative local search or statistical emulation approaches. The methodology herein thus provides a general framework for statistically robust local optimization. Our algorithm will almost always lead to a more robust solution than that obtained through local pattern search, and we have also observed that it can provide faster optimization in problems that are difficult for local methods.

Although we have focused on deterministic objective functions, the optimization algorithm is directly applicable to optimization in the presence of random error. Changes to the nugget parameter prior specification are all that is required for TGP to model noisy data, and APPS adapts for observation error by increasing the tolerance δ in the *sufficient decrease criterion* (refer to Section 2.1). Since the primary stopping criterion is based on step length, and step length decreases with each search that does not produce a new best point, δ ensures that evaluations are not wasted optimizing noise.

An appealing aspect of any oracle scheme is that, since points are given in addition to those generated by the pattern search, there is no adverse affect on the local convergence. In the setting of robust optimization, either with or without observation error, there are two additional major criteria by which convergence will be assessed. First, the converged solution should not lie on a *knifes edge* portion of the response surface. Second, the response at this solution needs to be *close* to the global optimum. In each case, the statistical emulator provides guidance as to the acceptability of any converged solution. Informally, the mean predictive surface allows the optimizer to judge the shape of the response around the solution and the magnitude of the optimum with respect to other potential optima around the input space. And the full accounting of TGP uncertainty provides a measure of the level of confidence that may be placed in this predicted surface. Formally, quantiles of predicted improvement are a precise measure for the risk of a significantly better alternate optimum at unobserved locations. For example, a 95th percentile for improvement $I(\boldsymbol{x})$ which is zero over the input domain would support a claim that the converged solution corresponds to a global optima. Future work in this direction could lead to substantial contributions in applied optimization.

References

Alexandrov, N., Dennis, J. E., Lewis, R. M., and Torczon, V. (1998). "A trust region framework for managing the use of approximation models in optimization." *Structural Optimization*, 15, 16–23.

Booker, A. J., Dennis, J. E., Frank, P. D., Serafini, D. B., Torczon, V., and Trosset, M. W. (1999). "A rigorous framework for optimization of expensive functions by surrogates." *Structural and Multidisciplinary Optimization*, 17, 1–13.

Chipman, H., George, E., and McCulloch, R. (1998). "Bayesian CART model search (with discussion)." *Journal of the American Statistical Association*, 93, 935–960.

— (2002). "Bayesian treed models." *Machine Learning*, 48, 303–324.

Cogdell, J. R. (1999). *Foundations of Electronics*. Prentice Hall.

Currin, C., Mitchell, T., Morris, M., and Ylvisaker, D. (1991). "Bayesian prediction of deterministic functions, with applications to the design and analysis of computer experiments." *Journal of the American Statistical Association*, 86, 953–963.

Fang, K.-T., Li, R., and Sudjianto, A. (2006). *Design and modeling for computer experiments*. Boca Raton: Chapman & Hall/CRC.

Fjeldy, T. A., Ytterdal, T., and Shur, M. S. (1997). *Introduction to device modeling and circuit simulation*. Wiley-InterScience.

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

Gramacy, R. B. (2007). "tgp: An R package for Bayesian nonstationary, semiparametric nonlinear regression and design by treed Gaussian process models." *Journal of Statistical Software*, 19.

Gramacy, R. B. and Lee, H. K. H. (2008). "Bayesian treed Gaussian process models with an application to computer modeling." *Journal of the American Statistical Association*, 103, 1119–1130.

Gray, G. A. and Kolda, T. G. (2006). "Algorithm 856: APPSPACK 4.0: Asynchronous parallel pattern search for derivative-free optimization." *ACM T. Math. Software*, 32, 3, 485–507.

Gray, G. A. et al. (2007). "Designing dedicated experiments to support validation and calibration activities for the qualification of weapons electronics." In *Proceedings of the 14th NECDC*. Also available as Sandia National Laboratories Technical Report SAND2007-0553C.

Green, P. (1995). "Reversible jump Markov chain Monte Carlo computation and Bayesian model determination." *Biometrika*, 82, 711–732.

Hedar, A.-R. and Fukushima, M. (2006). "Tabu Search directed by direct search methods for nonlinear global optimization." *European Journal of Operational Research*, 127, 2, 329–349.

Higdon, D., Kennedy, M., Cavendish, J., Cafeo, J., and Ryne, R. (2004). "Combining field data and computer simulations for calibration and prediction." *SIAM Journal of Scientific Computing*, 26, 448–466.

Jones, D., Schonlau, M., and Welch, W. (1998). "Efficient global optimization of expensive black-box functions." *Journal of Global Optimization*, 13, 455–492.

Keiter, E. R. (2004). "Xyce parallel elctronic simulator design: mathematical formulation." Sandia National Laboratories Technical Report SAND2004-2283.

Kennedy, M. and O'Hagan, A. (2001). "Bayesian calibration of computer models." *Journal of the Royal Statistical Society, Series B Statistical Methodology*, 63, 425–464.

Kolda, T. G. (2005). "Revisiting asynchronous parallel pattern search for nonlinear optimization." *SIAM J. Optimiz.*, 16, 2, 563–586.

Kolda, T. G., Lewis, R. M., and Torczon, V. (2003). "Optimization by direct search: new perspectives on some classical and modern methods." *SIAM Review*, 45, 385–482.

— (2006). "Stationarity results for generating set search for linearly constrained optimization." *SIAM J. Optimiz.*, 17, 4, 943–968.

Lee, H. K. H., Taddy, M., and Gray, G. A. (2008). "Selection of a representative sample." Tech. Rep. UCSC-SOE-08-12, University of California, Santa Cruz, Department of Applied Mathematics and Statistics. Also available as Sandia National Labs Report SAND2008-3857J.

McKay, M., Beckman, R., and Conover, W. (1979). "A comparison of three methods for selecting values of input variables in analysis of output from a computer code." *Technometrics*, 21, 239–245.

Mease, D. and Bingham, D. (2006). "Latin hyperrectangle sampling for computer experiments." *Technometrics*, 48, 467–477.

Morris, R. D., Kottas, A., Taddy, M., Furfaro, R., and Ganapol, B. (2008). "A statistical framework for the sensitivity analysis of radiative transfer models." *IEEE Transactions on Geoscience and Remote Sensing*, 12, 4062–4074.

Oakley, J. and O'Hagan, A. (2004). "Probabilistic sensitivity analysis of complex models: a Bayesian approach." *Journal of the Royal Statistical Society, Series B*, 66, 751–769.

O'Hagan, A., Kennedy, M. C., and Oakley, J. E. (1998). "Uncertainty analysis and other inference tools for complex computer codes." In *Bayesian Statistics 6*, eds. J. M. Bernardo, J. O. Berger, A. P. Dawid, and A. F. M. Smith, 503–524. Oxford University Press.

Regis, R. G. and Shoemaker, C. A. (2007). "Improved strategies for radial basis function methods for global optimization." *J. of Global Optimization*, 37, 1, 113–135.

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

Saltelli, A. (2002). "Making best use of model evaluations to compute sensitivity indices." *Computer Physics Communications*, 145, 280–297.

Saltelli, A., Chan, K., and Scott, E., eds. (2000). *Sensitivity analysis*. John Wiley and Sons.

Saltelli, A., Ratto, M., Andres, T., Campolongo, F., Cariboni, J., Gatelli, D., Saisana, M., and Tarantola, S. (2008). *Global sensitivity analysis: The primer*. John Wiley & Sons.

Santner, T., Williams, B., and Notz, W. (2003). *The design and analysis of computer experiments*. Springer-Verlag.

Schonlau, M., Welch, W., and Jones, D. (1998). "Global versus local search in constrained optimization of computer models." In *New Developments and Applications in Experimental Design*, eds. N. Flournoy, W. F. Rosenberger, and W. K. Wong, 11–25. Institute of Mathematical Statistics.

Sedra, A. S. and Smith, K. C. (1997). *Microelectronic Circuits*. 4th ed. Oxford University Press.

Stein, M. (1987). "Large sample properties of simulations using Latin Hypercube sampling." *Technometrics*, 143–151.

Wright, M. H. (1996). "Direct search methods: Once scorned, now respectable." In *Numerical Analysis 1995 (Proceedings of the 1995 Dundee Biennial Conference in Numerical Analysis)*, eds. D. F. Griffiths and G. A. Watson, vol. 344 of *Pitman Research Notes in Mathematics*, 191–208. CRC Press.