# Bayesian Guided Pattern Search for Robust Local Optimization

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

January 3, 2008

**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.

## 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 sim-

[*]M. Taddy is Ph.D. student and H. K. H. Lee is Associate Professor, Department of Applied Mathematics and Statistics, University of California, Santa Cruz, CA 95064, USA. G. A. Gray and J. D. Griffin are technical staff members of Sandia National Laboratories, Livermore, California. 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.

ulation. 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 for quick convergence, but use a statistical analysis of the entire function space to 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 et al., 2003; Higdon et al., 2004), to act as a guide for underlying local optimization. Our framework could thus be classified as an *oracle* optimization approach (see Kolda et al., 2003a, and references therein), wherein 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 et al., 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.

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 for the optimization 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. In Section 4, we illustrate our methods on our motivating example involving calibration of a circuit device simulator. Two simpler test problems are introduced below in the next section. Finally, in Section 5, we investigate convergence and begin to consider how statistical information can move beyond the search pattern and be used to tune the inherent parameters of pattern search optimization.

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

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

where $-10 \leq x_i \leq 10$ for $i = 1, 2$. The single 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)). Except when initial sampling is used to determine a starting location, the initial guess for each problem is $\mathbf{x} = (4, 4)$.

The response surfaces are shown in the background of Figure 1. Some particular difficulties

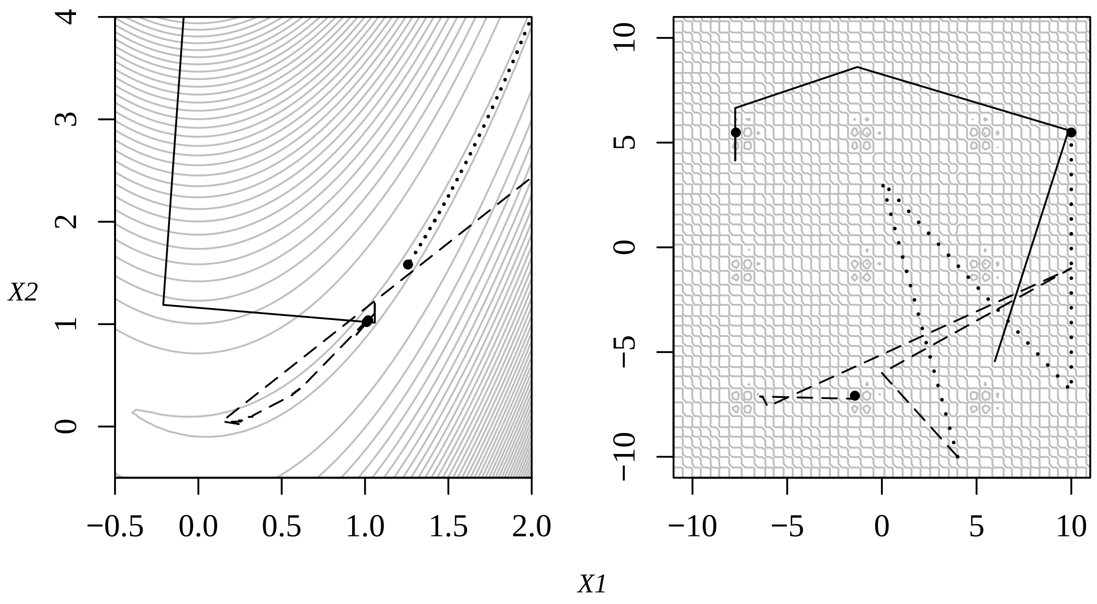

Figure 1: Rosenbrock (left) and Shubert (right) test problems. Contours for the response surface are plotted in grey, and trace paths for the *best point* during optimization are shown as a dotted line for APPS, a dashed line for APPS-TGP, and a solid line for APPS-TGP following an initialization through Latin hypercube sampling.

for optimization algorithms are emphasized in these problems. The Rosenbrock solution lies at the end of a long steep valley, which will typically cause problems for local search methods such as gradient descent and pattern search. As illustrated in the sensitivity analysis presented in Figure 5, the Shubert problem response surface variation is almost completely due to interaction between the two inputs rather than any marginal main effect. It is also evident, in Figure 1, that the Shubert problem involves rapid oscillations of the response surface around the larger scale response trends, thus presenting challenges that are similar to a truly noisy environment.

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 significantly more robust; in two of ten runs of the Shubert problem from the same starting location, APPS twice converged to minor solutions on the input boundaries, while TGP-APPS always found a global solution (APPS is non-deterministic due to random response time of the parallel processors). Such results are only magnified as the complexity of the problem increases, as illustrated in the circuit application of Section 4.

## 2 ELEMENTS OF THE METHODOLOGY

The optimization algorithm consists of 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.

### 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 falls into the category of direct search methods (Kolda et al., 2003b). Direct search denotes a branch of derivative-free optimization algorithms that make no attempt to explicitly evaluate, estimate, or model local derivatives. They thus tend to be more robust for difficult optimization problems (those which may be noisy, with occasional nonsmooth, discontinuous, or undefined points), than derivative-based approaches that break down at points where the derivative ceases to exist.

The APPS algorithm is a good deal 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 (2006a). At each iteration of APPS, three basic steps are executed:

1. generate a set of trial points $Q_k$,
2. send trial points $Q_k$ to the compute cluster, and obtain a set of function evaluations $R_k$,
3. update current *best point* $\mathbf{x}_k$.

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. In simple pattern search, trial points are generated using a positive spanning set of search directions $D = \{d_1, \ldots, d_\ell\}$ and have the form $Q_k = \{\mathbf{x}_i + \Delta d_i \mid 1 \leq i \leq \ell\}$, for a positive step size $\Delta$. 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. Convergence to locally optimal points is ensured using a *sufficient decrease criterion* for accepting new best points. A trial point $\mathbf{x}_k + \Delta d_i$ is considered a better point if $f(\mathbf{x}_k + \Delta d_i) - f(\mathbf{x}_k) < \delta$, for user defined $\delta > 0$. For the unconstrained case, using standard assumptions, it can be shown that

$$\lim \inf_{k \to \infty} \|\nabla f(\mathbf{x}_k)\| \to 0.$$

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

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 (2006b) and references therein. The latest software is publicly available under the terms of GNU L-GPL at `http://software.sandia.gov/appspack/`.

## 2.2 Treed Gaussian process emulation

A Bayesian approach was brought to the emulation of computer code in the paper by Currin et al. (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 et al., 1989; O'Hagan et al., 1998; Fang et al., 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.

Treed Gaussian process (TGP) models form a natural extension of this methodology and provide a more flexible nonstationary regression scheme (Gramacy and Lee, 2008). These 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 GP models 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 *et al.* (1998 & 2002), using 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. Further details of implementation and properties for TGP are available in 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` ). In fact, all of the statistical methods described in this paper are available with version 2.0 of `tgp`.

The TGP model involves a complex prior specification in order to promote mixing over alternative tree structures. However, the prior parameterization 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). In particular, the stationary GPs allow for a linear mean trend and are assumed to have an anisotropic Gaussian correlation function,

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

Here, $d$ is the dimension of the input space and each range parameter $\theta_k$ 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 $\sigma^2$ the GP variance and $\gamma$ the nugget parameter. The nugget term requires the only non-default parameterization, 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, $\gamma$ is forced to be small through the prior specification $\pi(\gamma) = \text{gamma}(1, 100)$. The presence of a small nugget allows for smoothing the predicted response surface to avoid the potential instability that is inherent in point-by-point interpolation (demonstrated in the contrasting surface fits of Figure 2). 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. We note that the nugget also allows for the possibility of numerical instability in real-life supposedly deterministic 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

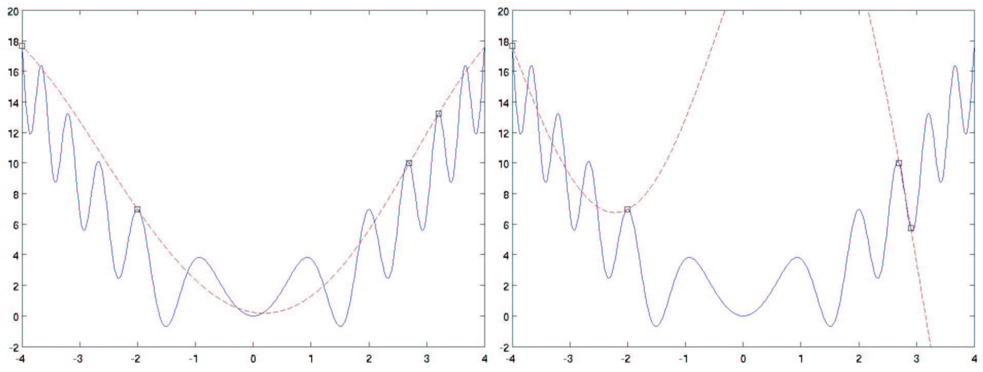

Figure 2: If two points are allowed to become too close together for highly wavy function, interpolation methods can break down. The dashed lines show the instability of a thin-plate spline fit to four observations from the underlying function.

are shown in Figure 3 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.

## 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 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 et al., 1999; Alexandrov et al., 1998). The Expected Global Optimizer (EGO) algorithm developed by Jones et al. (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

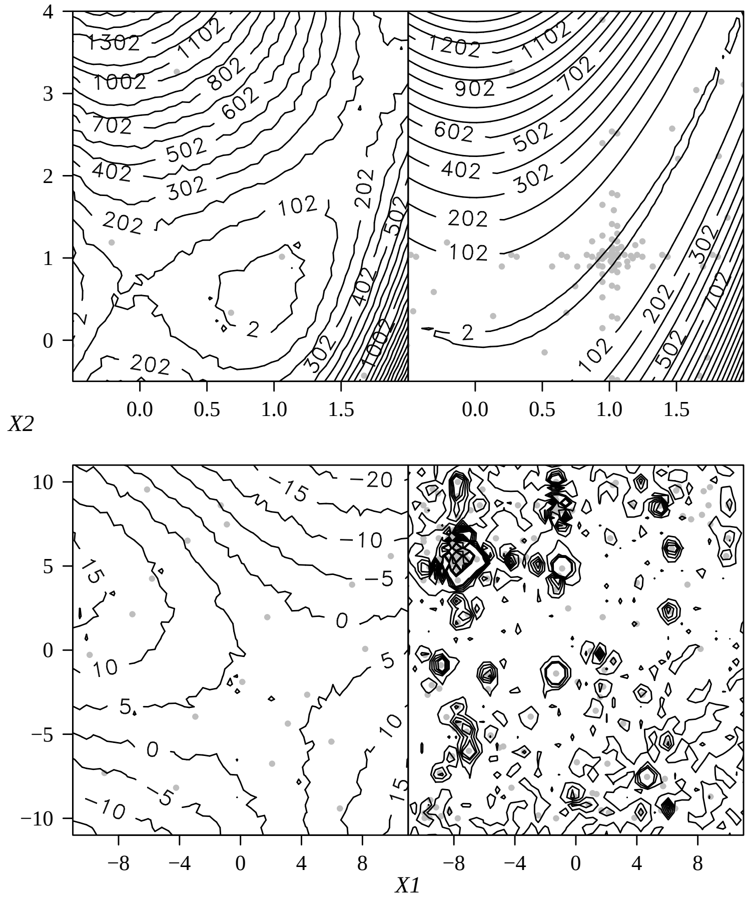

Figure 3: Posterior mean predictive response surfaces for Rosenbrock (top) and Shubert (bottom) problems. The left hand column corresponds to fit conditional on the initial sample of 20 points, and the right hand column corresponds to TGP fit to all of the iterations at the time of convergence (128 for Rosenbrock, 260 iterations for Shubert). In each case, the evaluated iterates are plotted in grey.

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, i.e. robust local optimization through pattern search, 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 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 the efficient parallel implementation of such hybrid algorithms.

## 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 $I(\mathbf{x}) = \max\{f_{best} - f(\mathbf{x}), 0\}$, where $f_{best}$ is response corresponding to the present *best point* in the search (as defined above in Section 2.1). 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 best 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) 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 $I^g(\mathbf{x}) = \max\{(f_{best} - f(\mathbf{x}))^g, 0\}$, 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 $\mathbb{E}[I^0(\mathbf{x})] = \Pr(I(\mathbf{x}) > 0)$, $g = 1$ yields the standard statistic, and $g = 2$ explicitly rewards the improvement variance since $\mathbb{E}[I^2(\mathbf{x})] = \text{var}[I(\mathbf{x})] + \mathbb{E}[I(\mathbf{x})]^2$. Our examples employ both $g = 1$ and $g = 2$, but this is a tuning parameter that may be altered for particular implementations. We have experienced some success with a $g$ that varies throughout the optimization, decreasing as the process converges. 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 4 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 $g$ parameterizations, however this can lead to substantially different search patterns based on our ranking algorithm.

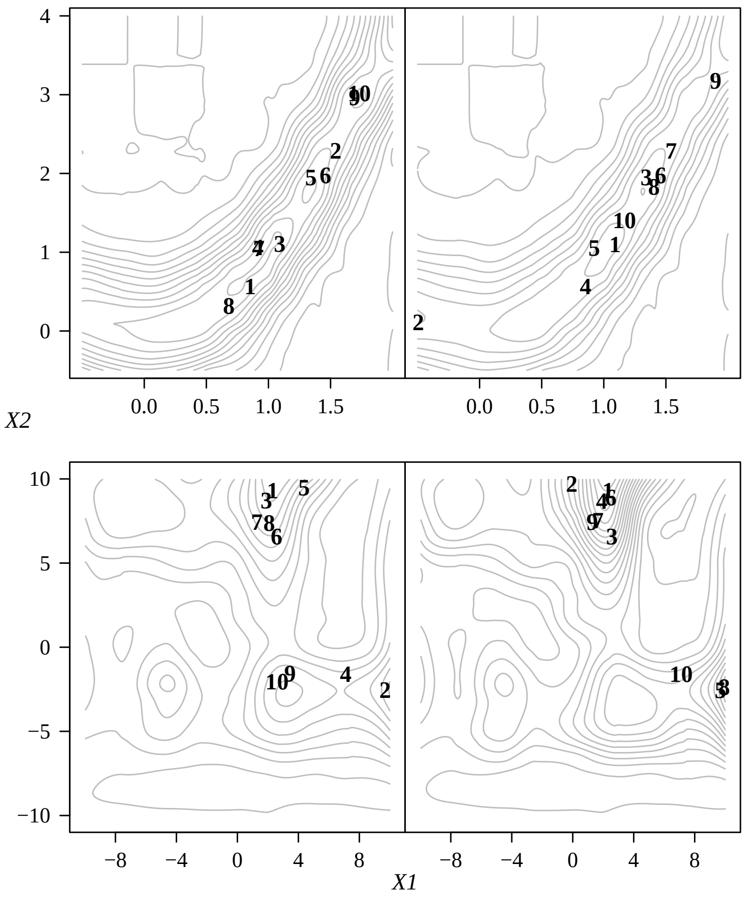

Figure 4: Expected improvement surfaces $\mathbb{E}[I^g(\mathbf{x})]$ and ranks of the top ten candidate locations, for the Rosenbrock (top) and Shubert (bottom) problems, conditional on TGP fit to 75 function evaluations. The left hand plots correspond to $g = 1$ and the right hand plots to $g = 2$.

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

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

(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 input space onto a dense candidate set $\tilde{\mathbf{X}}$ of $M$ locations. Although optimization over this set will not lead to 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 $I(\tilde{\mathbf{x}}_i)$ 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 and predicted response) to obtain the posterior sample $\{I^g(\tilde{\mathbf{X}})_1, \ldots, I^g(\tilde{\mathbf{X}})_T\}$. We then proceed recursively to build an *ordered* search pattern of $m$ locations: Designate $\mathbf{x}_1 = \mathrm{argmax}_{\tilde{\mathbf{x}} \in \tilde{\mathbf{X}}} \mathbb{E}\left[I^g(\tilde{\mathbf{x}})\right]$, 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

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

Thus, after each $j$th additional point is added to the set, we have the maximum expected $j$-location improvement. 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 4. 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 et al., 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 ball 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. With the uncertainty distribution uniform over the bounded input space, this approach was used for the applications throughout this paper.

### 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 the 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, 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, the initial sample of function evaluations may be used to define parameters of the optimization. In particular, sensitivity analysis (SA) can be performed to resolve the sources of output variability by apportioning elements of this variation to different sets of input variables (Saltelli et al., 2000). 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.

SA comes in a variety of different forms, depending on the goal of the analysis. In the context of optimization, it is often local sensitivity analysis that is utilized – the effect of small perturbations to converged solutions is investigated to avoid *knifes edge* areas of the response surface. We are more interested in the global sensitivity analysis that is required at the beginning of an optimization, wherein the variability of the objective is investigated with respect to the uncertainty distribution over the entire input space and sensitivity is always considered in relation to the uncertainty density $u$. Two influential sensitivity indices, which will be useful in this setting, are the first-order for the $j$th 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 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

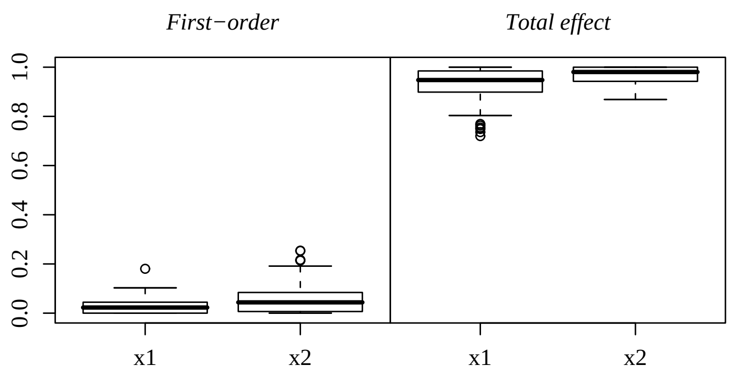

Figure 5: Shubert problem. Sensitivity analysis based on a TGP fit to an LHS of 20 initial locations, summarized by first order sensitivity indices (left) and total sensitivity indices (right).

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 Saltelli et al. (2000) for a complete analysis framework.

Estimation of these sensitivity indices usually requires a large number of function evaluations in order to obtain Monte Carlo estimates for the necessary integrals. 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. Indeed, using the Monte Carlo estimation procedure of Saltelli (2002) based upon prediction from the TGP statistical emulator, conditional on an initial sample of function evaluations at 20 input locations, we obtain posterior samples of the first-order and total sensitivity indices for the Shubert problem with respect to a uniform uncertainty distribution over the bounded input region. At each iteration of the MCMC model fitting, response predictions were drawn over a set of input locations generated as prescribed in Saltelli's scheme, and estimates for the indices were calculated based upon this predicted response. The results, in Figure 5, clearly show that interaction between the two input variables is responsible for most of the variability in the Shubert function response. 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 et al. (2007) 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 run 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.

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 with conveyor. This process continues until either a maximum budget of evaluations has been reached or all citizens

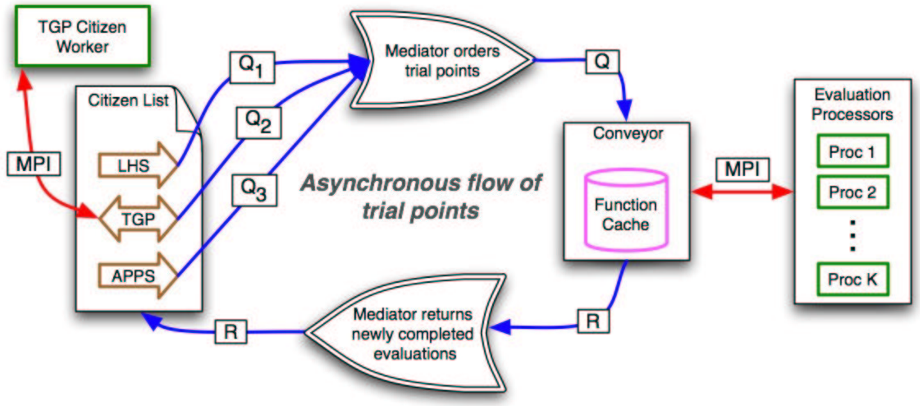

Figure 6: Schematic illustration of the hybrid parallel implementation for TGP-APPS.

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 stores a function value cache that lexicographically stores a history of all completed function evaluations using a splay tree as described in Gray and Kolda (2006a); 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.

Within our algorithm, the LHS citizen is given highest priority and hence has all points evaluated first. Because TGP is slower to submit points than APPS (TGP performs an MCMC analysis of the current function evaluation history off-line), 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

| Method | Problem | Evaluations | Solution |
|---|---|---|---|
| APPS | Rosenbrock | 14709 | 6.64e-2 |
| APPS-TGP (no LHS) | Rosenbrock | 554 | 1.02e-4 |
| APPS-TGP (with LHS) | Rosenbrock | 128 | 3.67e-4 |
| APPS | Shubert | 28 | -48.51 |
| APPS-TGP (no LHS) | Shubert | 295 | -186.73 |
| APPS-TGP (with LHS) | Shubert | 260 | -186.73 |

Table 1: Rosenbrock and Shubert problem results. For each optimization algorithm, the number of function evaluations required and objective response at converged solution.

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).

Figure 6 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.

## 3.4 Results for example problems

The results from optimization on the test problems, using stand-alone APPS as well as the hybrid TGP-APPS, both with and without an initial Latin hypercube sample of 20 points, are illustrated in Figure 1 and in Table 1. For the Rosenbrock problem, we see that APPS required a vast increase in computational expense over APPS-TGP methods and was unable to locate the global minimum. In the case of the Shubert problem, we present an unsuccessful APPS run which converged to a minor solution on the input boundary, while the TGP-APPS methods converged to global solutions (albeit at an increased computational expense). We

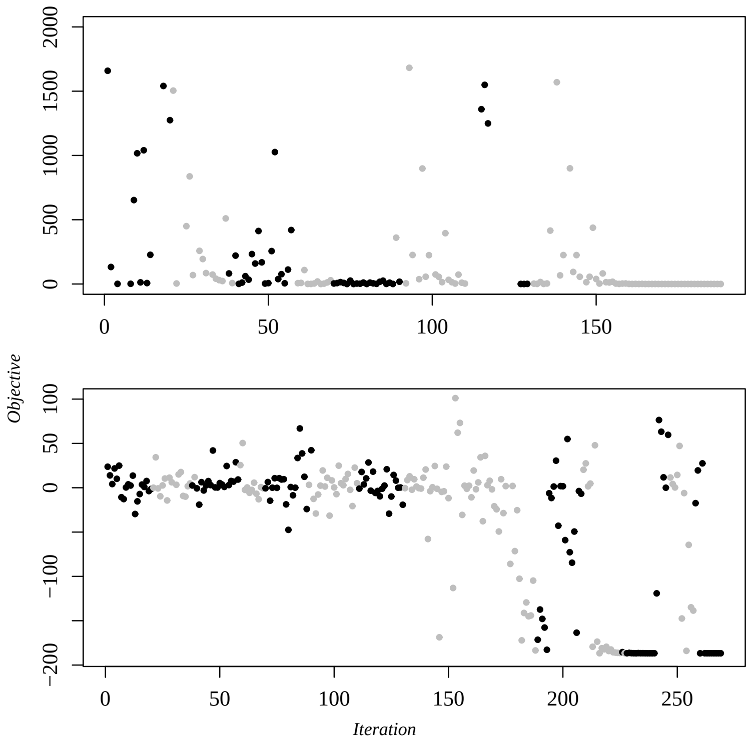

Figure 7: Objective function response trace during optimization of the Rosenbrock (top) and Shubert (bottom) problems. Black points were generated by the TGP ranking algorithm, and grey points correspond to the APPS search pattern.

note, however, that eight out of ten runs of the APPS algorithm were able to locate the global solution in between 80 and 140 iterations, thus showing an improvement in efficiency over the more robust TGP-APPS. The failed APPS run is presented here to highlight the risk involved with efficient local methods that take no global search into account.

Figure 7 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 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.

## 4 OPTIMIZATION OF A TRANSISTOR SIMULATOR

We now discuss optimization of a radiation-aware simulation model for an electrical device component of a circuit. The goal of the optimization is to find simulator tuning parameter values that lead to simulator output that is as close as possible (under the distance function defined in equation (5), below) to results from real-world experiments involving the electrical devices of interest. The model input in this case 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). All bjts share a basic underlying model and 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, consist of six observations per device obtained at three different testing facilities. In each experiment, the devices of interest are exposed to a unique radiation photocurrent 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).

The particular simulator of interest is a Xyce implementation of the Tor Fjeldy photocurrent model for the bjt. Xyce is an electrical circuit simulator developed at Sandia National Laboratories (Keiter, 2004), and the Tor Fjeldy photocurrent model is described in detail in Fjeldy et al. (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

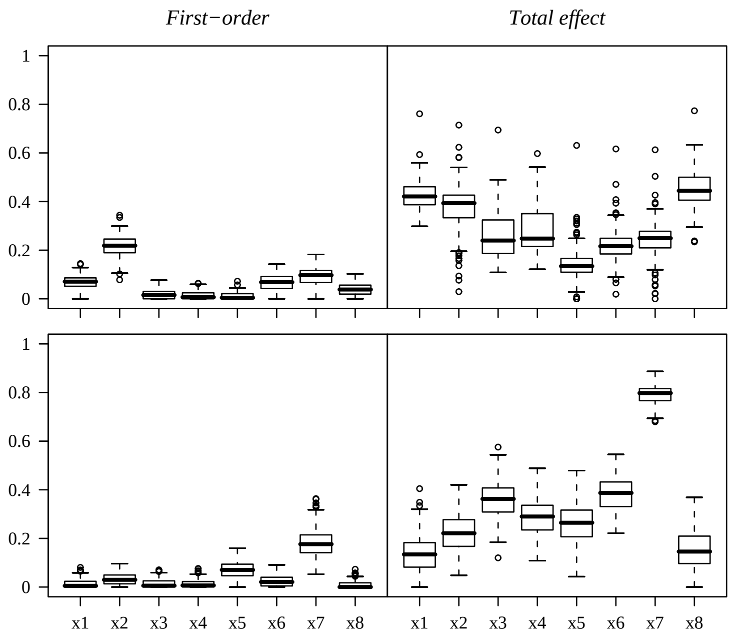

Figure 8: Sensitivity analysis for *bft92a* (top) and *bfs17a* (bottom) devices, based on a TGP fit to an LHS of 160 initial locations, summarized by first order sensitivity indices (left) and total sensitivity indices (right).

a set of specific radiation pulse inputs:

$$f(\mathbf{x}) = \sum_{i=1}^{N} \frac{1}{T_i} \sum_{t=1}^{T_i} \left[ (S_i(t;\mathbf{x}) - E_i(t))^2 \right]. \tag{5}$$

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 $\mathbf{x}$. These parameters include 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. Indeed, Figure 8

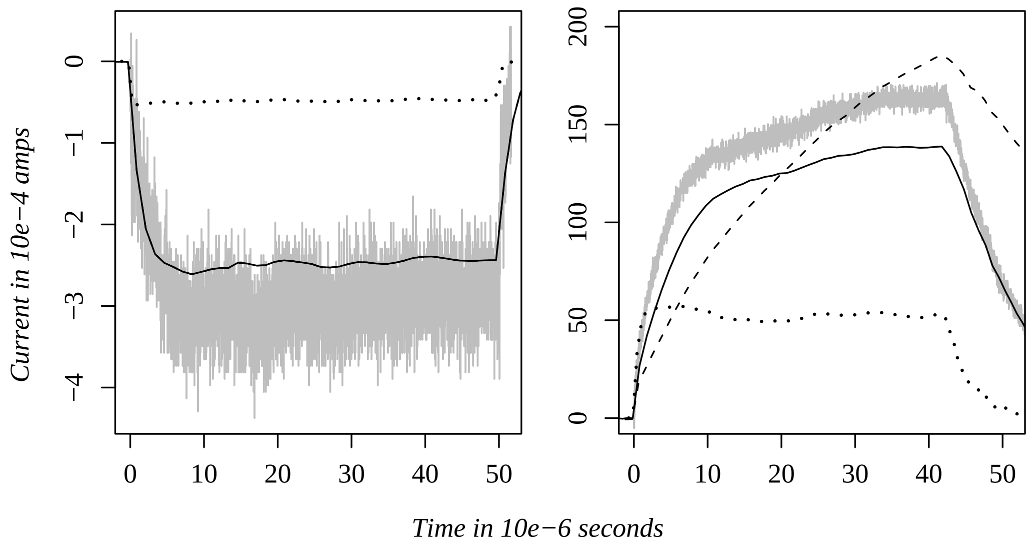

Figure 9: Simulated response current curves, corresponding to different tuning parameter values, for the *bft92a* (left) and *bfs17a* (right) devices. The solid line shows response for parameters found using TGP-APPS, the dashed line for parameters found through APPS alone, and the dotted line for the initial parameter vector guess. The experimental current response curves for the radiation impulse used in these simulations is shown in grey.

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, and all of the eight parameters appear to have significant effect on the objective function variability.

The objective of (5) was optimized using both APPS and the hybrid algorithm TGP-APPS. In the case of the hybrid algorithm, initial LHS of 160 points were used to inform the TGP. The wall clock time and the number of objective function evaluations corresponding to each device and each optimization algorithm are shown in Table 2. Figure 9 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. 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 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 pat-

| Method | Device | Evaluations | Time |
|---|---|---|---|
| APPS | *bft92a* | 6823 | 341257 sec $\approx$ 95 hrs |
| APPS-TGP | *bft92a* | 962 | 49744 sec $\approx$ 14 hrs |
| APPS | *bfs17a* | 811 | 37038 sec $\approx$ 10 hrs |
| APPS-TGP | *bfs17a* | 1389 | 65122 sec $\approx$ 18 hrs |

Table 2: For each bjt device and each optimization algorithm, the number of objective function evaluations and total wall clock time required to find a solution.

tern quickly into decent areas of the input space. We note that that a similarly low computational time (56815 seconds $\approx$ 16 hours) was required to obtained 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 reasonable 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. Note that the time results support this, as they show that the APPS algorithm converged relatively quickly. The extra computational cost of TGP-APPS is well justified by the improvement in fit.

For each device, the optimization solutions obtained through TGP-APPS provide a reasonable match between the response simulated by Xyce and the true experimental data. Although a perfect match was unobtainable due to unresolved uncertainty and bias in the photocurrent model, both the modelers and experimentalists were pleased with these results. 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.

# 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 pattern search or statistical emulation approaches. The methodology herein thus provides a general framework for statistically robust local optimization.

We close with a discussion of how the work may be expanded into post-processing and analysis of convergence. One 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. But it is also possible to have the statistical model actually inform the convergence properties. Our algorithm is based on the statistical generation of search patterns, either in initial sampling or through prediction from a statistical emulator. There exists, however, potential for extension of this approach to allow the statistical analysis to influence the parameters of the underlying local pattern search algorithm. The optimization of noisy objective functions provides for one such extension. Although we have focused on deterministic objective functions, the optimization algorithm is directly applicable to optimization in the presence of random error; since only changes to the nugget parameter prior specification are required for TGP to model noisy data, the presented methodology will apply so long as the underlying local search method is able to accommodate noise in the response.

APPS deals with observation error by setting the tolerance $\delta$ for the *sufficient decrease criterion*, used in determination of new best points (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, $\delta$ ensures that evaluations are not wasted optimizing noise. Choosing $\delta$, however, requires advance knowledge of the response surface. The TGP posterior response distributions will provide an estimate of the observation error (through the nugget parameter), and this can be used to determine an appropriate tolerance level. Indeed, since none of the

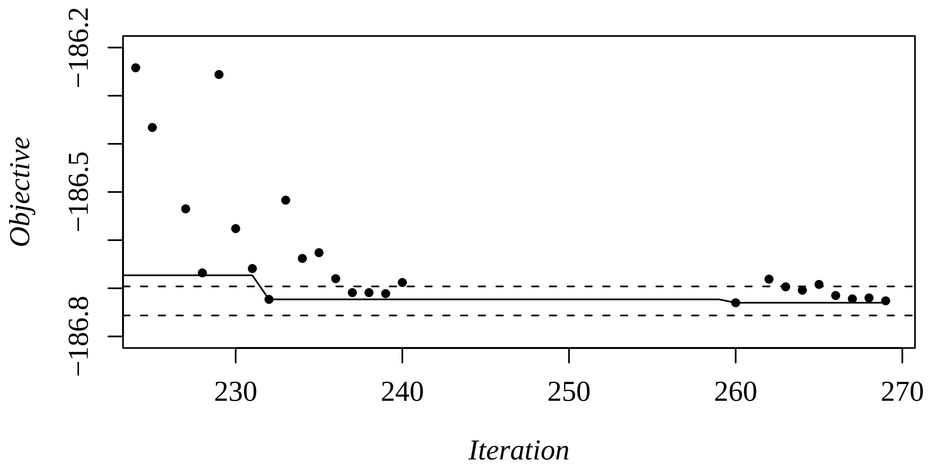

Figure 10: The Shubert test objective during convergence. The solid line shows the best point trace, and the dashed lines are the 5th and 95th posterior percentiles for observation error, from TGP fit to all 260 function evaluations, around the solution. The vertical axis is limited to response values close to the converged solution, and thus not all iterations appear.

assumptions in the convergence proof of Kolda and Torczon (2004) require constant tolerance, $\delta$ can be adapted during the optimization without adversely affecting local convergence. Figure 10 shows the final iterations of a Shubert problem optimization using the hybrid TGP-APPS algorithm. Recall that the Shubert function response surface involves rapid oscillations that present many of the same properties for optimization as random noise. A TGP fit after convergence to the entire set of 260 function evaluations, with a gamma$(1, 1)$ prior on the nugget $\gamma$ to allow for noise, provides an estimate of observation error around the converged solution. We see that the $\delta$ used in this optimization would have been too small if the objective was to be treated as random, as function evaluations were wasted searching for improvement within the range for observation error.

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. What constitutes a knifes edge and how close to global the solution needs to be are application specific. But 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. Formal quantification is also possible. For example, quantiles of predicted improvement are a precise measure for the risk of a significantly better alternate optimum and could even be used as the basis for additional stopping criteria.

## 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.

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, 9.

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*. To appear.

Gray, G. A. and Kolda, T. G. (2006a). "Algorithm 856: APPSPACK 4.0: Asynchronous Parallel Pattern Search for Derivative-Free Optimization." *ACM T. Math. Software*, 32, 3, 485–507.

— (2006b). "Algorithm 856: APPSPACK 4.0: Asynchronous Parallel Pattern Search for Derivative-Free Optimization." *ACM TOMS*, 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." Tech. Rep. SAND2004-2283, Sandia National Labs, Albuquerque, NM.

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. (2003a). "Optimization by direct search: new perspectives on some classical and modern methods." *SIAM Review*, 45, 385–482.

— (2003b). "Optimization by direct search: new perspectives on some classical and modern methods." *SIAM Rev.*, 45, 3, 385–482.

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

Kolda, T. G. and Torczon, V. (2004). "On the convergence of asynchronous parallel pattern search." *SIAM Journal on Optimization*, 14, 939–964.

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. (2007). "A statistical framework for the sensitivity analysis of radiative transfer models used in remote sensed data product generation." Tech. rep., Dept. of Applied Math & Statistics, University of California, Santa Cruz.

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.

Santner, T., Williams, B., and Notz, W. (2003). *The Design and Analysis of Computer Experiments*. Springer-Verlag.

Schonlau, M., Jones, D., and Welch, W. (1998). "Global versus local search in constrained optimization of computer models." In *New Developments and applications in experimental design*, no. 34 in IMS Lecture Notes - Monograph Series, 11–25.

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.