Pattern Search Optimization with a Treed Gaussian Process Oracle

Matt Taddy[1], Genetha A. Gray[2], Herbert K. H. Lee[1], Robert B. Gramacy[3], Monica Martinez-Canales[2]
Department of Applied Mathematics & Statistics, University of California, Santa Cruz[1]
Computational Sciences & Mathematics Research Department, Sandia National Laboratories[2]
The Statistical Lab, University of Cambridge[3]

February 17, 2007

Abstract

This work combines pattern search optimization with a statistical emulator based on Treed Gaussian Processes (TGP) to create a new hybrid algorithm. The goal is to use the global probabilistic view provided by TGP to inform the local pattern search and form a more intelligent optimization algorithm. We also propose ways in which the emulator can be used to gain information about the objective function, inform the algorithm stopping rules and provide a probabilistic analysis of the type of convergence. We present the algorithm, a framework for statistically informed optimization, and illustrate the work with numerical results.

Key words: derivative-free optimization; expected improvement; Gaussian processes; partitioning;

1 Background and motivation

Significant advances in computing capabilities, decreasing storage costs, and the rising expenses associated with physical experiments have contributed to an increase in the use of numerical modeling and simulation. In line with this trend, simulation-based optimization has become a common tool in the design and control of complex engineering systems. Often, the simulation itself is quite complex and researchers have sought to use less expensive surrogate functions, in conjunction with results from the simulator, to guide the optimization. Many variations of this type of approach can be found in the literature (Alexandrov *et al.*, 1998; Nash, 2000, for example).

We expand on these ideas by fitting a statistical model as an emulator for the computer function, and use the posterior distribution for the output surface as a guide for optimization. There is a rich literature in the analysis of complex computer models and building statistical emulators. Our approach in this aspect is to use stochastic process priors for the simulation function, and this places the methodology firmly within the tradition of existing computer experiments literature. The novelty of this work lies in a hybridization – we combine the "global" scope of the statistical emulator with the "local" pattern search optimization to construct a new hybrid algorithm.

The local optimization method, Asynchronous Parallel Pattern Search (APPS), and the emulator model, based on Treed Gaussian Processes (TGP), are outlined in Sections (1.1) and (1.2) respectively. Under the proposed scheme, the *improvement* statistic from the emulator is used to choose additional locations for evaluation within the pattern search. Section (2) describes the specifics of the new algorithm resulting from a hybridization of TGP and APPS. Numerical results are presented in Section (2.2).

Inclusion of a statistical emulator is valuable beyond its abilities to suggest points for the pattern search. Indeed, the emulator provides a complete quantification of uncertainty with respect to the objective function. This can be used to make intelligent decisions about the state of the optimization, and provides a framework for sensitivity analysis. In section (2.1) we propose using information from the emulator as part of the stopping criteria, and in section (3) we develop a framework for sensitivity analysis in an optimization setting. These steps provide a global scope to the stopping rules, and allow one to make probabilistic statements about the robustness of a converged solution.

1.1 Pattern Search optimization

We consider a derivative-free optimization method called Asynchronous Parallel Pattern Search (APPS) (Hough *et al.*, 2001). The APPS algorithm is part of a class of direct search methods which were primarily developed to address problems in which the derivative of the objective function is unavailable and approximations are unreliable (Wright, 1996). Pattern searches use a predetermined pattern of points to sample a given function domain. It has been shown that if certain requirements on the form of the points in this pattern are followed and if the objective function is suitably smooth, convergence to a stationary point is guaranteed (Torczon, 1997). The APPS algorithm is an asynchronous version of PPS that reduces processor latency and requires less total time than standard PPS to return results (Hough *et al.*, 2001). Implementations of APPS have minimal requirements on the number of processors and do not assume that the amount of time required for an objective function evaluation is constant or that the processors are homogeneous. Our specific APPS algorithm is described in Kolda (2004). It is provably convergent under mild conditions (Kolda and Torczon, 2004). Omitting the implementation details, the basic APPS algorithm can be simply outlined as follows:

1. Generate a set of trial points and evaluate them.
2. Process the set of evaluated points and see if it contains a new *best point*. If there is such a point, then the iteration is successful; otherwise, it is unsuccessful.
3. If the iteration is successful, replace the current best point with the new best point. Optionally, regenerate the set of search directions and delete any pending trial points in the conveyor.
4. If the iteration is unsuccessful, reduce certain step lengths as appropriate. In addition, check for convergence based on the step lengths.

A complete mathematical description and analysis of this algorithm is available in Kolda (2004). 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.

1.2 Statistical emulation

The role of a statistical emulator in this context is to add robustness and introduce global properties to the local optimization method. To accomplish these goals, we use ideas from the design

and analysis of computer experiments literature and use stochastic process priors to model the deterministic computer output function.

The standard practice in the computer experiments literature is to model the output of the simulations as a realization of a stationary Gaussian Process (GP) (Sacks *et al.*, 1989; Santner *et al.*, 2003; Fang *et al.*, 2006). The treed Gaussian process (TGP) models form a natural extension of this methodology and provide a more flexible nonstationary regression scheme. 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 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 (CART) work of Chipman *et al.* (1998 & 2002) , namely through a set of operations on a recursive partition tree. These operations – prune, grow, swap, change, and rotate – are proposed through a reversible jump Markov chain Monte Carlo (RJMCMC) algorithm (Green, 1995), and the tree is fit simultaneously with the parameters of the individual GP models. In this way, all parts of the model can be learned automatically from the data and the Bayesian model averaging through RJMCMC allows for explicit estimation of predictive uncertainty. For the independent GP, we allow for a linear trend and, again following the standard in the literature, we use the Gaussian correlation function. There are different range parameters in each dimension. Further details of implementation and properties for TGP are outlined extensively in Gramacy's thesis (Gramacy, 2005), as well as in the technical report by Gramacy and Lee (Gramacy and Lee, 2006). There is software available in the form of a `tgp` library for the open source statistical package R. (see `http://www.cran.r-project.org/src/contrib/Descriptions/tgp.html`).

2 Hybridization with an Oracle

In recent years, some optimization methods have introduced an *oracle* to predict points at which a decrease in the objective function might be observed. These points are given in addition to the iterates inherent to the optimization method itself. Analytically, an oracle is free to choose points by any finite process. (See Kolda *et al.* (2003) and references therein.) The addition of an oracle is particularly amenable to a pattern search methods like APPS, since the iterate(s) suggested by the oracle are merely additions to the pattern. Furthermore, the asynchronous nature of the APPS implementation makes it adept at handling the evaluation of additional points.

We use the TGP statistical emulator as our oracle. Each time that the oracle is called, we use RJMCMC to fit the TGP model to the existing set of evaluated iterates $(x_i, f(x_i))$. Thus, after each oracle call we have a posterior predictive sample of output values over a set of candidate locations, conditional on the points that have already been evaluated, and any statistic depending upon this output is easily obtained.

The expected improvement at a point x, $\mathbb{E}[I(x) = \max(f(x) - f_{max}, 0)]$, is a useful criterion for choosing new locations for evaluation. The paper by Jones *et al.* (1998) illustrates the use of this statistic in an optimization scheme. Note that the improvement is always non-negative, and does not penalize locations in proportion to the magnitude of a negative $f(x) - f_{max}$. Suppose that the simulator is run at two locations x_1 and x_2, and that $f(x_1) < f(x_2) < f_{max}$. The non-negative improvement statistic is a rational reflection of belief that the run at x_2 is no more useful that the run at x_1 – in each case we do not have a new point, and each run provides information about the output surface.

Since the improvement is a random variable, this criteria will reward points where the output is highly uncertain, as well as where the function is generally predicted to be better than the present best point. In our hybrid algorithm, the expected improvement is calculated for each posterior predictive output set. The points with highest expected improvement are passed back to the APPS algorithm for evaluation. These points are then evaluated to see if one of them is a new best point. If not, the point is merely stored with the data. However, if a TGP point is a new best point, the APPS search pattern continues from this location. In practice the oracle may suggest any number of points each time it is called. In our work here, a single point is suggested but we are not claiming that this is optimal.

For implementation of the above algorithm, it is necessary to choose a set of candidate locations over which we predict the output and the improvement. The fitting of the TGP is not substantially slowed by increasing the size of the candidate set over which we are predicting output. Thus if the set of potential inputs is discrete and not too large ($>> 1000$), it is reasonable to include this entire grid for prediction at each run of the oracle. If, however, the potential input space is continuous, we clearly need another solution. In the optimization of Jones *et al.* (1998), they use a branch and bound algorithm to choose new locations for evaluation. We have previously experimented with D-optimal designs which take into account the already evaluated iterates, but in contrast with standard "experiment design" scenario, space filling is not our primary objective here. Rather, since the evaluated iterates were already chosen, more or less, for their potential as optima, we are likely to require further sampling close to these locations. In the end, the best solution we have found is to draw a Latin hypercube design of workable size each time the oracle is called, ignoring the location of existing iterates, and use this as the candidate set. This seems to work well in practice.

Throughout all of this, APPS is optimizing as it normally would, except with a search pattern augmented with points chosen by the emulator oracle. In particular, the stopping rules for the optimization are provided by APPS alone and the vast amount of information provided by the emulator is used only in the suggestion of new points. In the following section, we propose a first step towards a more informed optimization with a stopping rule based upon information from the statistical emulator.

2.1 A statistical convergence criterion

One of the main theoretical considerations in developing an optimization method like the one described here is convergence. In this case, the APPS algorithm is provably convergent, to a local optimum, under mild conditions. Since the oracle points are given in addition to those generated by the pattern search, there is no adverse affect on the convergence. We would like to do better, and as a first step into tapping the information provided by a statistical oracle, we devise a statistical criterion for global convergence.

In standard APPS, the primary stopping condition is based on the step length. This criterion was chosen because it can be shown that if the objective function is continuously differentiable, then the norm of the gradient can be bounded as multiple of the step size (Kolda *et al.*, 2003). Alternatively, APPS offers two additional stopping criteria. One is based on whether or not the function has reached a specified threshold, and the other is defined in terms of the number of function evaluations.

Our additional stopping condition depends upon the predicted improvement, a statistic that is calculated over the set of candidate locations each time the oracle is called. Recall that the

improvement, $I(x) = \max(f(x) - f_{max}, 0)$, is already sampled and that the posterior expectation is used to choose new iterates. Our convergence criteria will concentrate on a quantile, rather than the mean, improvement. A risk level p and a numerical tolerance level ϵ are chosen. Each time the TGP emulator is fit, we will take the $(1-p)$th quantile for the posterior sample from $I(x)$ at each x candidate location. If the quantile for I is greater than ϵ anywhere over candidate set, then the optimization is not allowed to converge and more points are sampled until either a new maximum is found, or the probability of finding an alternate optimum decreases below p. This will ensure that the TGP oracle does not stop if there is non-ignorable probability of finding a better optimum at one of the candidate locations. In the work presented below, we have used $p = 0.05$ for a tolerance $\epsilon = 0.01$.

2.2 Example

In practice, we have implemented these ideas and tested them with some promising results. The example presented here involves a two dimensional test problem. The equation for this function, shown in Figure (1), is $f(x, y) = -g(x)g(y)$ where

$$g(z) \quad = \quad \exp\left(-(z-1)^2\right) + \exp\left(-0.8(z+1)^2\right) - 0.05\sin\left(8(z+0.1)\right)$$

The starting guess was $(0.2, 0.3)$. Without the oracle, APPS converges to a local maximum of -1.060 at $x = [1.125, 1.153]$. We can see in Figure (2) that, for a TGP emulator fit to evaluated iterates at the time of convergence, the mean predicted surface is a poor approximation to the true output and large variations in expected improvement are observed over the input space. The continuous portion of the posterior distribution for $I([2, -2])$ is also included in this plot, illustrating the shape of our uncertainty about this quantity. Note that there is also a point mass $P(I([2, -2]) = 0) = 0.9525$. The algorithm has already converged to a local maximum and there is little room for improvement, a point relfected in the large probability of $I = 0$.

The oracle enhanced version of APPS converged to a maximum of -1.126 at $x = [-1.055, -1.043]$, which is within tolerance of the global solution. Figure (3) illustrates the information provided by the oracle during the hybrid optimization process. In the plot for the oracle fit during optimization, we can see that the expected improvement surface contains considerable information to guide the oracle in suggesting a new point for evaluation. The surface for 95^{th} percentile improvement also shows significant probability of finding a better f_{max} than the present best point. Conversely, in the bottom plot illustrating oracle fit after convergence, both the expected and 95^{th} percentile improvement surfaces are flat over the input space. Indeed, our convergence criteria has required that the 95^{th} percentile improvement be less than 0.01 at all 500 candidate locations.

3 Sensitivity analysis

Inclusion of a statistical convergence criterion is only beginning to tap the full potential of information available in the TGP oracle. One could further utilize the TGP model by performing a sensitivity analysis whenever the oracle is called. Since the RJMCMC fit is already required by the hybrid algorithm, this analysis could be done in parallel at little or no extra cost.

A full sensitivity analysis of the sort outlined in Oakley and O'Hagan (2004) could provide the user with a probabilistic map for how the variability of their objective function changes over the input space, which would allow them to make informed decisions about reasonable input ranges

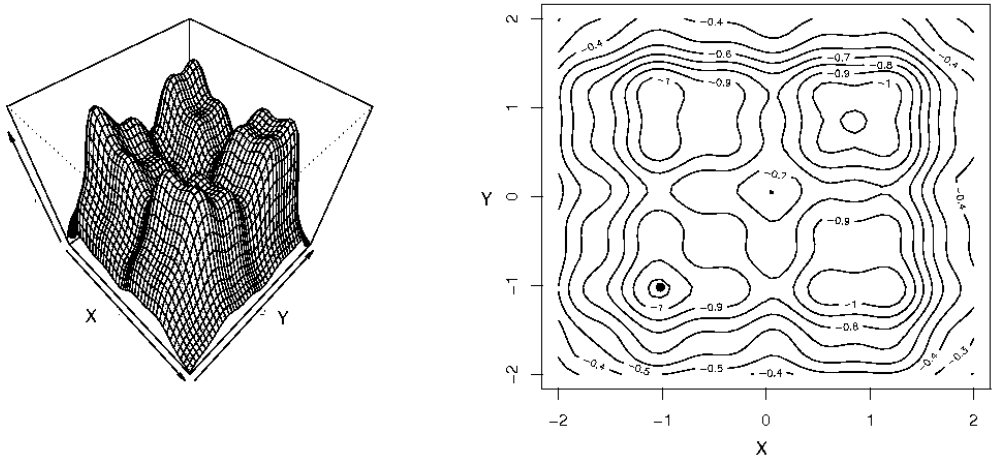

Figure 1: The function used for the numerical testing (left) and its contour lines (right). The true maximum, at $x_1 = -1.0408$ $x_2 = -1.0408$, is marked on the plot.

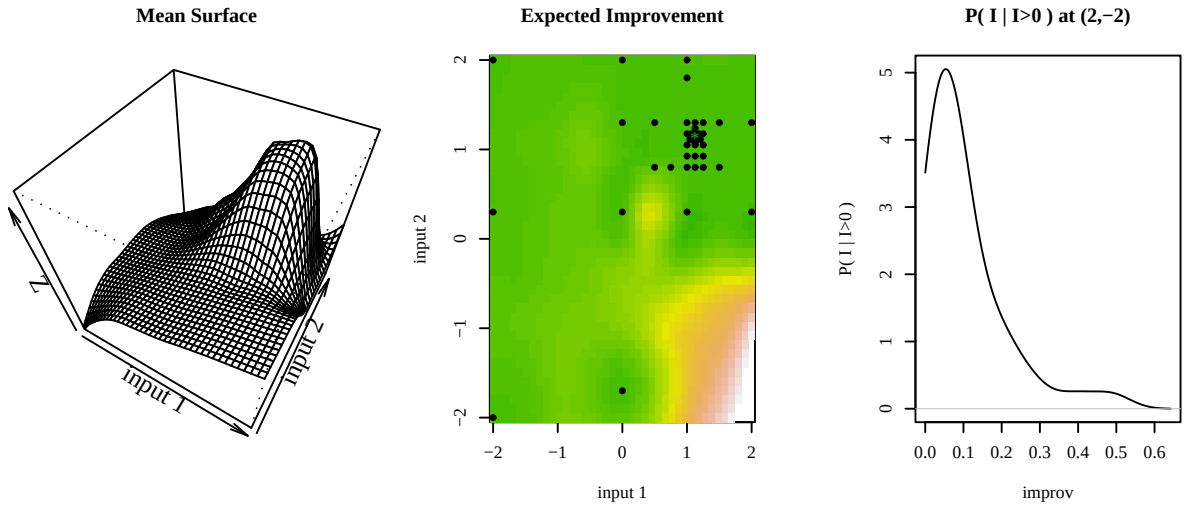

Figure 2: **Non-Oracle APPS** Surfaces from a TGP fit to the evaluated iterates from a non-oracle APPS optimization; from left to right, the mean predicted output surface and the expected improvement (rising from green to white) over the input space, followed by the posterior distribution for the positive improvement at a single point (the corner $[-2, 2]$). The black points are evaluated iterate locations, and the green star is the point of convergence

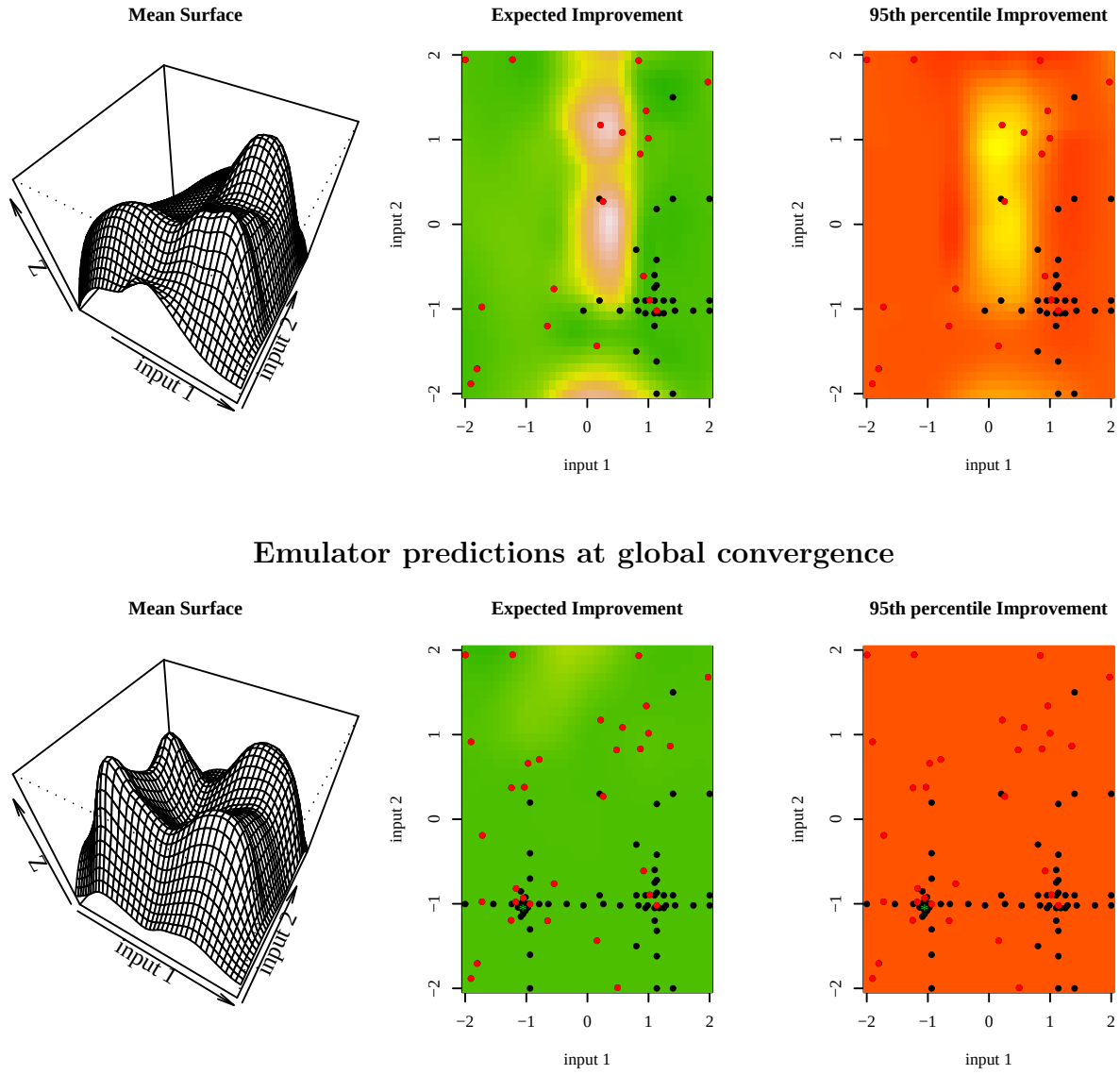

Figure 3: **APPS-TGP hybrid** These pictures show some of the information provided by the TGP oracle, at different points in the optimization process. The top plots are from an oracle fit during the optimization, and the bottom plots are from an emulator fit after convergence to the global maximum. In each case, from left to right, we see mean predicted output surface, the expected improvement (rising from green to white), and the 95th percentile for improvement (rising from red to white) over the input space. The black dots are points chosen by APPS, the red dots by the TGP oracle, and the green star is the point of convergence.

and model adequacy. We will discuss here a single aspect that is especially applicable in the optimization setting. Local sensitivity analysis is the study of how output changes over small input neighborhoods. This information can help optimizers to avoid the pitfalls of unstable convergence and "knifes edge" optima.

When the derivative of the objective function is not reliably approximated, as is common for complex computer models, the statistical emulator can be used to provide a local sensitivity analysis. Suppose that the converged optimal solution is found at input location x_o. It is often the case that the true input to the real system is uncertain and distributed in some way around the nominal input, which here would be x_o if this solution is to be applied. If we call this error distribution $G_o(x)$, then one metric which could be useful is the variance of our expected output

$$V_o = \mathrm{var}\left[\mathbb{E}_{\mathrm{G_o}}[\mathrm{f(X)|X}]\right] = \int_{\mathbb{R}^{\mathrm{d}}} \left(\mathbb{E}[\mathrm{f(w)|w}] - \mathbb{E}_{\mathrm{G_o}}[\mathrm{f(X)}]\right)^2 \mathrm{dG_o(w)}.$$

Recall that f is a random function wherever the simulator has not been run. This quantity will provide a measure of robustness for any converged solution. In order to incorporate this into the TGP oracle framework, we use a Monte Carlo estimate for V_0. After the optimization has converged and an input error distribution G_o centered on the solution has been specified, a group of locations $\{x_{oi}\}$ are drawn from G_o. The TGP model is then fit to the evaluated iterates and function output is predicted over $\{x_{oi}\}$, with the variance of the posterior predictive sample used as an estimate of V_0. When we wish to input our solution into a real system, this number can be used to decide whether the optimum is robust enough for the intended application. As a quick illustration, when this is applied to the synthetic example above, we have a global maximum at $x_o = [-1.055, -1.043]$. Postulating an input error distribution that is $\mathrm{N}(x_o, 0.05^2\mathrm{I})$, we estimated $\hat{V}_o = 0.00012$.

4 Discussion

The hybrid algorithm, statistical convergence criterion, and local sensitivity analysis introduce some of the ideas that need to be incorporated for optimization algorithms to be used as decision makers. For a next-generation algorithm, the statistical emulator should be able to exert more of an influence on the optimization while it is in process. A complete parallel sensitivity analysis would eventually allow the user to dismiss subsets of the domain that exhibit large variances or that exceed critical thresholds. And information from the oracle could provide more comprehensive answers to the optimization problem. For example, a regional optimum could be used to generate a robust set containing multiple optima from which the designer can choose; a solution which would be welcome when faced with the design issues of real systems. Finally, future algorithms must assess simulation costs so that iterates are only evaluated if they meet a set computational budget. In ongoing work with simulation-based optimization at Sandia National Laboratories, we are working to expand the present scheme to include some of the above ideas.

The use of TGP as our model for the statistical emulator makes for a fast implementation and a very flexible model, and we believe it will lead to a superior analysis in many situations. However, the ideas of hybridization with a statistical emulator and of a convergence criteria based upon posterior predictions are not dependent upon the type of model used. Similarly, there is potential for hybridization with other deterministic optimization schemes, although the APPS algorithm is very nicely suited to such applications. Thus we hope that the use of a statistical emulator to provide probabilistic snapshots of the optimization will be popular with those working in this area, regardless of the particular model choice.

References

Alexandrov, N., Jr., J. E. D., Lewis, R. M. and Torczon, V. (1998) A trust region framework for managing the use of approximation models in optimization. *Structural Opt.*, **15**.

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

Chipman, H., George, E. and McCulloch, R. (2002) Bayesian treed models. *Machine Learning*, **48**, 303–324.

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

Gramacy, R. B. (2005) *Bayesian Treed Gaussian Process Models*. Ph.D. thesis, University of California, Santa Cruz, CA 95064. Department of Applied Math & Statistics.

Gramacy, R. B. and Lee, H. K. H. (2006) Bayesian treed Gaussian process models. Tech. rep., Dept. of Applied Math & Statistics, University of California, Santa Cruz.

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

Green, P. (1995) Reversible jump markov chain monte carlo computation and bayesian model determination. *Biometrika*, **82**, 711–732.

Hough, P. D., Kolda, T. G. and Torczon, V. (2001) Asynchrounous parallel pattern search for nonlinear optimization. *SIAM J. Sci. Comput.*, **23**, 134–156.

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

Kolda, T. G. (2004) Revisiting asynchronous parallel pattern search. Tech. Rep. SAND2004-8055, Sandia National Laboratories, Livermore, CA 94551.

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.

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

Nash, S. G. (2000) Multigrid approach for discretized optimizaton problems. *J. Opt. Methods and Software*, **14**, 99–116.

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.

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

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

Torczon, V. (1997) On the convergence of pattern search algorithms. *SIAM J. Opt.*, **7**, 1–25.

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.