Gaussian Processes for Global Optimization

Michael A. Osborne, Roman Garnett, and Stephen J. Roberts

Department of Engineering Science
University of Oxford
Oxford, UK OX1 3PJ
{mosb,rgarnett,sjrob}@robots.ox.ac.uk

Abstract. We introduce a novel Bayesian approach to global optimization using Gaussian processes. We frame the optimization of both noisy and noiseless functions as sequential decision problems, and introduce myopic and non-myopic solutions to them. Here our solutions can be tailored to exactly the degree of confidence we require of them. The use of Gaussian processes allows us to benefit from the incorporation of prior knowledge about our objective function, and also from any derivative observations. Using this latter fact, we introduce an innovative method to combat conditioning problems. Our algorithm demonstrates a significant improvement over its competitors in overall performance across a wide range of canonical test problems.

Key words: Gaussian Processes, Bayesian Methods, Decision Theory, Global Optimization, Noisy Optimization, Non-convex Optimization

1 Introduction

Optimization has been an important area of mathematical study since the creation of the digital computer. A great deal of research has been devoted to solving dozens of sub-problems in this field [1], with application areas ranging from economics to mechanical design. In this paper we approach global optimization from the viewpoint of Bayesian theory, and frame the problem as a sequential decision problem. Imagine that we have an unknown and expensive-to-evaluate objective function $y(x)$ that we seek to minimize. The cost associated with computing $y(x)$ compels us to select the location of each new evaluation very carefully. For testing purposes, we reflect this cost by restricting ourselves to a limited number of function evaluations. Our problem's ultimate task is to return a final point x_M in the domain, and we define the loss associated with this choice, $\lambda(x_M)$, to be $y(x_M)$. As we aim to minimize our losses, we aim to minimize $y(x_M)$.

Using a Gaussian process framework, we derive an analytic expression for the expected loss of evaluating $y(x)$ at a given candidate point under a limited myopic approximation. The resulting algorithm is similar to a previous proposal [2] employing Gaussian processes (under the name "kriging"). However, we can improve further upon this criterion by considering multiple function evaluations into the future. Our clear Bayesian formalism also permits us to benefit from the

specification of the required confidence in our returned value. Further, our use of Gaussian processes allows us to incorporate useful prior information about the objective function (such as periodicity), as well as learning from observations of the derivative. This final fact also leads to an innovative resolution of conditioning issues. We conclude by presenting the extension of our method to the optimization of noisy functions.

2 Gaussian Processes

Gaussian processes (GPs) offer a powerful method to perform Bayesian inference about functions [3]. This inference is at the heart of optimization, made explicit by techniques of optimization that employ response surfaces or surrogates [4]. Our goal is to build a statistical picture of the function's overall form given our observations of it.

A GP is defined as a distribution over the infinite number of possible outputs of our function, such that the distribution over any finite number of them is multivariate Gaussian. It is completely specified by a mean vector $\boldsymbol{\mu}$ and a covariance matrix $\mathbf{K}$:

$$p(\,\boldsymbol{y} \mid \boldsymbol{x},\, I\,) \triangleq \mathrm{N}(\boldsymbol{y}; \boldsymbol{\mu}, \mathbf{K}) \triangleq (\det 2\pi\mathbf{K})^{-\frac{1}{2}} \exp\left(-0.5\,(\boldsymbol{y}-\boldsymbol{\mu})^{\mathrm{T}}\,\mathbf{K}^{-1}\,(\boldsymbol{y}-\boldsymbol{\mu})\right) . \tag{1}$$

Here I, the *context*, includes prior knowledge of both mean and covariance functions. For testing, we take a constant prior mean μ and the sum of two isotropic squared exponential covariance functions:

$$K(x, x') \triangleq h_A^2 \exp\left(-r_A^2/2\right) + h_B^2 \exp\left(-r_B^2/2\right) , \tag{2}$$

where, if d indexes the dimensions of the input space, $r_A^2 \triangleq \sum_d (x_d - x'_d)^2/w_A^2$ is a non-periodic term and $r_B^2 \triangleq \sum_d (\sin \pi(x_d - x'_d)/w_B)^2$ a periodic term. This is intended to allow us to model functions that are the superposition of a non-periodic and a periodic component. Fortunately, of course, there exist a wide variety of other covariance functions should the problem call for them. Note also that we need not place the GP on the function directly—clearly, a function known to be strictly positive might benefit from a GP over its logarithm. Other transformations might always be considered to improve our performance for a particular function.

We collectively denote our hyperparameters as θ. This includes the output scales h_A, h_B and input scales w_A, w_B, along with the constant prior mean μ. Define I_0 as the conjunction of I and whatever observations of the function we have gathered so far, $(\boldsymbol{x}_0, \boldsymbol{y}_0)$. Taking both I_0 and θ as given, we are able to write our predictive equations for $\boldsymbol{y}_\star$ at $\boldsymbol{x}_\star$

$$p(\,\boldsymbol{y}_\star \mid \boldsymbol{x}_\star,\, \theta,\, I_0\,) = \mathrm{N}\big(\boldsymbol{y}_\star; \boldsymbol{m}_\theta(\boldsymbol{y}_\star|I_0), \mathbf{C}_\theta(\boldsymbol{y}_\star|I_0)\big) , \tag{3}$$

where we have:

$$\begin{aligned} \boldsymbol{m}_\theta(\boldsymbol{y}_\star|I_0) &= \boldsymbol{\mu}_\theta(\boldsymbol{x}_\star) + \mathbf{K}_\theta(\boldsymbol{x}_\star, \boldsymbol{x}_0)\mathbf{K}_\theta(\boldsymbol{x}_0, \boldsymbol{x}_0)^{-1}(\boldsymbol{y}_0 - \boldsymbol{\mu}_\theta(\boldsymbol{x}_0)) \\ \mathbf{C}_\theta(\boldsymbol{y}_\star|I_0) &= \mathbf{K}_\theta(\boldsymbol{x}_\star, \boldsymbol{x}_\star) - \mathbf{K}_\theta(\boldsymbol{x}_\star, \boldsymbol{x}_0)\mathbf{K}_\theta(\boldsymbol{x}_0, \boldsymbol{x}_0)^{-1}\mathbf{K}_\theta(\boldsymbol{x}_0, \boldsymbol{x}_\star) . \end{aligned} \tag{4}$$

Of course, we can rarely be certain about θ *a priori*. These hyperparameters must hence be marginalized. Although the required integrals are non-analytic, we can efficiently approximate them by use of Bayesian Monte Carlo [5] techniques. This entails evaluating our predictions for a range of hyperparameter samples $\{\theta_i : i \in S\}$, with a different mean $\boldsymbol{m}_i(\boldsymbol{y}_\star|I_0)$ and covariance $\mathbf{C}_i(\boldsymbol{y}_\star|I_0)$ for each, which are then combined in a weighted mixture

$$\begin{aligned} p(\,\boldsymbol{y}_\star \mid \boldsymbol{x}_\star,\, I_0\,) &= \frac{\int p(\,\boldsymbol{y}_\star \mid \boldsymbol{x}_\star, \theta, I_0\,)\, p(\,\boldsymbol{y}_0 \mid \boldsymbol{x}_0, \theta, I\,)\, p(\,\theta \mid I\,)\,\mathrm{d}\theta}{\int p(\,\boldsymbol{y}_0 \mid \boldsymbol{x}_0, \theta, I\,)\, p(\,\theta \mid I\,)\,\mathrm{d}\theta} \\ &\simeq \sum_{i \in S} \rho_i\, \mathrm{N}\big(\boldsymbol{y}_\star; \boldsymbol{m}_i(\boldsymbol{y}_\star|I_0), \mathbf{C}_i(\boldsymbol{y}_\star|I_0)\big)\,, \end{aligned} \tag{5}$$

with weights $\boldsymbol{\rho}$ as detailed in [6]. We use also the sequential formulation of a GP given by [6], a natural fit for our sequential decision problem. After each new function evaluation, we can efficiently update our predictions in light of the new information received.

3 Optimization Using Gaussian Processes

Given the GP model of our expensive function, we now determine where best to evaluate it in future.

3.1 One-step Lookahead

To begin, imagine that we have only one allowed function evaluation remaining before we must report our inferred function minimum. We constrain the returned (x_M, y_M) to the set of actual gathered points. Essentially, we require that the final reported minimum supplied by our algorithm must be known with the utmost confidence (although we will look at slightly relaxing this constraint later). Suppose $(\boldsymbol{x}_0, \boldsymbol{y}_0)$ are the function evaluations gathered thus far and define $\eta \triangleq \min \boldsymbol{y}_0$. Given this, we can define the loss of evaluating the function this last time at x and its returning y

$$\lambda(y) \triangleq \begin{cases} y; & y < \eta \\ \eta; & y \geq \eta \end{cases}. \tag{6}$$

That is, our loss is simply the new observed minimum $\min(y, \eta)$, which we would report as y_M.

Now, given our GP over y, we can determine an analytic form for the expected loss of selecting x given that we know I_0 and have only one evaluation remaining

$$\Lambda_1(\,x \,|\, I_0\,) \triangleq \int \lambda(y)\, p(\,y \mid x,\, I_0\,)\,\mathrm{d}y = \sum_{i \in S} \rho_i \mathrm{V}_i(\,x \,|\, I_0\,)\,, \tag{7}$$

where, denoting the usual Gaussian cumulative distribution function as Φ,

$$\begin{aligned} \mathrm{V}_i(\, x \mid I_0 \,) &\triangleq \eta \int_{\eta}^{\infty} \mathrm{N}(y; m_i, C_i)\ \mathrm{d}y + \int_{-\infty}^{\eta} y\, \mathrm{N}(y; m_i, C_i)\ \mathrm{d}y \\ &= \eta + (m_i - \eta)\, \Phi(\eta; m_i, C_i) - C_i\, \mathrm{N}(\eta; m_i, C_i)\ . \end{aligned} \tag{8}$$

Here we have abbreviated $m_i(y|I_0)$ as m_i and $C_i(y|I_0)$ as C_i. Note that the "expected improvement" function of [2] is close to but differs from this Bayesian expected loss criterion.

The location where our expected loss is lowest gives the optimal location for our next function evaluation. Note that (8) will decrease as m_i becomes lower than η, and also as C_i increases. This first fact ensures exploitation, the second exploration. As such, the minimization of our expected loss gives a very natural balancing of these two concerns.

Of course, we have merely shifted the minimization problem from one over the objective function $y(x)$ to one over the expected loss function $\Lambda_1(\, x \mid I_0\,)$. Fortunately, the expected loss function is continuous and computationally inexpensive to evaluate, as are the analytic expressions for its gradient and Hessian. We hence have a range of routines suitable for this minimization.

3.2 Multi-step Lookahead

Note that the situation described in Section 3.1 can also be used as a myopic approximation to the case in which we actually have many evaluations remaining. Essentially, this means we assume that we are so uncertain about how this current decision will affect our future decisions that we can ignore them. Such an approximation typically gives good results, as we'll see later.

Nonetheless, proper probabilistic reasoning will allow us to relax this shortsighted assumption. We define our successive remaining function evaluations as $\{(\boldsymbol{x}_j, \boldsymbol{y}_j) : j = 1, \ldots, n\}$. The totality of our information up to and including the jth evaluation we denote using I_j, the conjunction of I_0 and $\{(\boldsymbol{x}_{j'}, \boldsymbol{y}_{j'}) : j' = 1, \ldots, j\}$. In general, we use $\Lambda_n(\, x_1 \mid I_0\,)$ to denote the expected loss of selecting x_1 given that we know I_0, considering n evaluations into the future:

$$\begin{aligned} &\Lambda_n(\, x_1 \mid I_0\,) \triangleq \\ &\int \lambda(y_n)\, p(y_n|x_n, \theta, I_{n-1})\, p(x_n|I_{n-1}) \ldots p(y_2|x_2, \theta, I_1)\, p(x_2|I_1)\, p(y_1|x_1, \theta, I_0) \\ &\quad \mathrm{d}y_1 \ldots \mathrm{d}y_n \mathrm{d}x_2 \ldots \mathrm{d}x_n\, p(\boldsymbol{y}_0|\boldsymbol{x}_0, \theta, I)\, p(\theta|I)\, \mathrm{d}\theta \big/ \textstyle\int p(\boldsymbol{y}_0|\boldsymbol{x}_0, \theta, I)\, p(\theta|I)\, \mathrm{d}\theta \\ &\qquad = \textstyle\sum_{i \in S} \rho_i \int \mathrm{V}_i(\, x_n \mid I_{n-1}\,)\, \delta\big(x_n - \mathrm{argmin}_{x_\star}\, \Lambda_1(\, x_\star \mid I_{n-1}\,)\big) \ldots \\ &\qquad\quad \mathrm{N}\big(y_2; m_i(y_2|I_1), C_i(y_2|I_1)\big)\, \delta\big(x_2 - \mathrm{argmin}_{x_\star}\, \Lambda_{n-1}(\, x_\star \mid I_1\,)\big) \\ &\qquad\qquad \mathrm{N}\big(y_1; m_i(y_1|I_0), C_i(y_1|I_0)\big) \mathrm{d}y_1 \ldots \mathrm{d}y_{n-1} \mathrm{d}x_2 \ldots \mathrm{d}x_n\ . \end{aligned} \tag{9}$$

The probabilistic model underlying (9) is illustrated in Figure 2. To evaluate (9), we successively sample y_1 through y_{n-1} from their respective Gaussians, $\mathrm{N}\big(y_1; m_i(y_1|I_0), C_i(y_1|I_0)\big)$ through $\mathrm{N}\big(y_{n-1}; m_i(y_{n-1}|I_{n-2}), C_i(y_{n-1}|I_{n-2})\big)$.

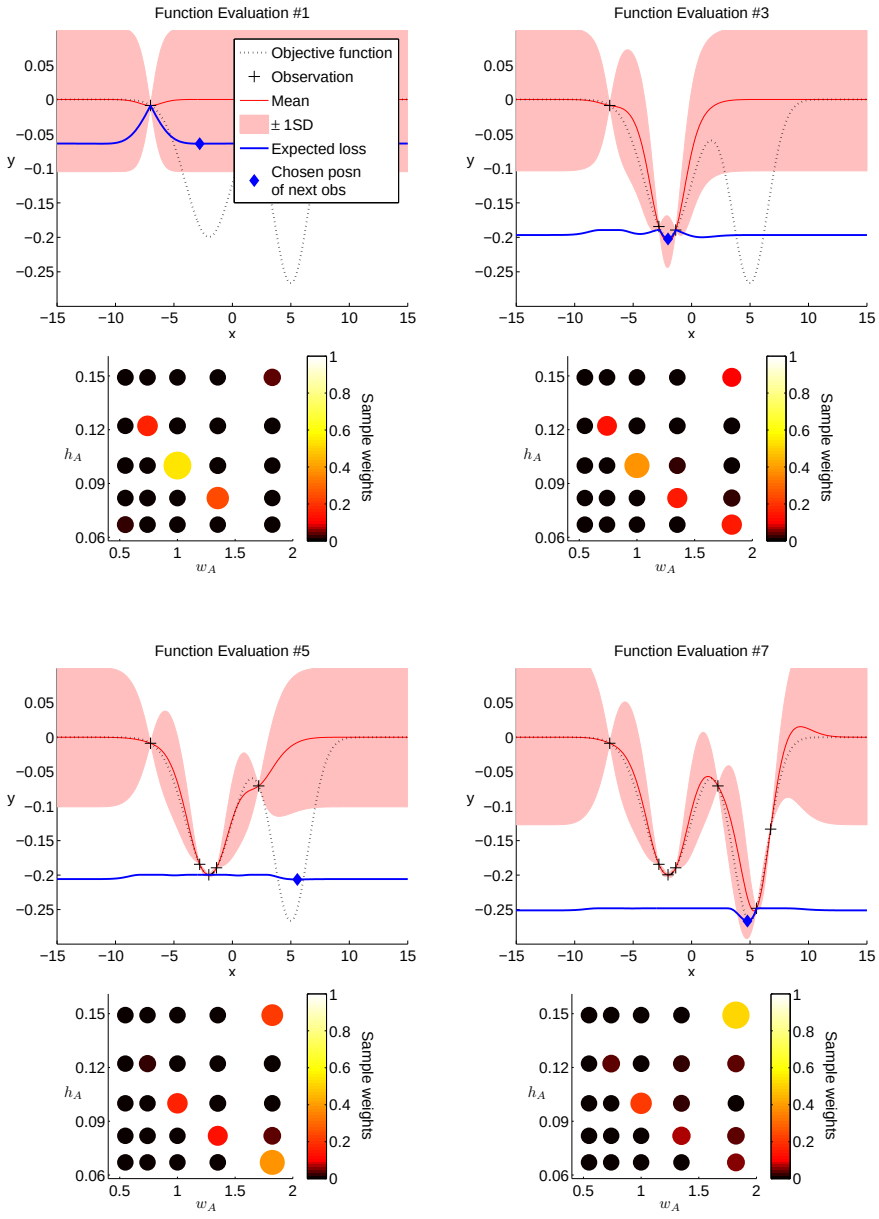

Fig. 1: An illustration of our algorithm balancing the exploitation of the current local minimum and exploration of unvisited areas. The chosen covariance was (2), with $h_B = 0$, giving us two scale hyperparameters, w_A and h_A. The weights associated with the samples over those hyperparameters are indicated both by the shading and the radii of the circles. Weight is transferred from the prior mean, at $w_A = 1$ and $h_A = 0.1$, towards larger and more representative scales. This gives rise to a smoother GP fit with larger error bars, which then influences the selection of future evaluations.

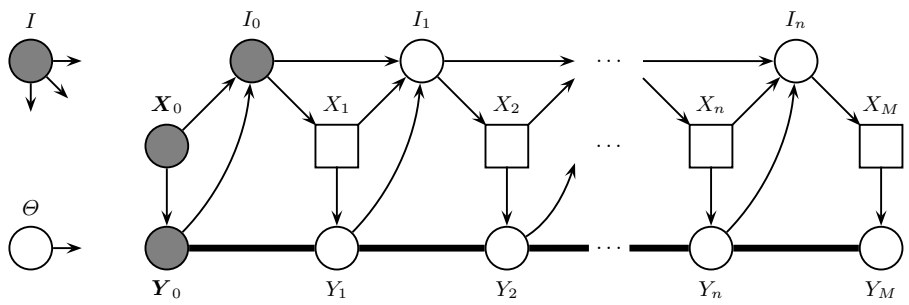

Fig. 2: A Bayesian network for the optimization problem. Shaded nodes are known, and square nodes represent the results of a decision. All Y nodes are correlated with one another, Θ is correlated with all Y nodes and the context I is correlated with all nodes.

Given each, we can then determine the best choice for the next sample location by minimizing the appropriate expected loss function $\Lambda_{n-j}(\, x_\star \,|\, I_j \,)$, for $j = 1, \ldots, n-1$. Note these loss functions are themselves integrals over θ. These will be approximated as weighted sums over hyperparameter samples θ_i, with weights that will differ from the ρ_i given the differing sets of information I_j. Unfortunately, only for $\Lambda_1(\, x_\star \,|\, I_{n-1} \,)$ can this required minimization benefit from analytic derivatives. Given this fact, and the degree of sampling required, it is clear that multi-step lookahead is a more computationally demanding exercise. However, this cost can be justified if the function $y(x)$ we aim to minimize is itself even more computationally expensive. We effectively need merely to select the n appropriate for the problem at hand.

3.3 Conditioning Problems

A common problem with GPs is the poor conditioning of covariance matrices. This occurs when we have observations that are strongly correlated (giving multiple rows/columns in the covariance matrix that are very similar) such as is the case for highly proximate observations. This is a particular concern for optimization, in which we commonly want to take many observations close to a minimum. Of course, this problem is ameliorated slightly by our procedure, which recognizes that sampling close to an existing observation rarely produces much new information. The problem is also less severe in high dimension, in which there is volume enough to avoid existing observations.

Rather than inverting the covariance matrix directly, we use its Cholesky decomposition. This gives us an improved degree of tolerance to conditioning problems. Use of a Moore–Penrose pseudoinverse, in which small singular values are zeroed in the singular value decomposition, is also possible, but is prohibitively slow to compute. We use a simple heuristic to guarantee good conditioning. Each new function observation is required to be sufficiently far from all existing function observations; the separation δ we enforce is a single input scale.

3.4 Derivative Observations

It is trivial to incorporate derivative observations into a GP [3], as a function over which we have a GP is jointly Gaussian with all of its derivatives and anti-derivatives. This property means that derivative observations can be treated just as observations of the function itself, albeit with a modified covariance function. Hence any observations of the derivatives or anti-derivatives of our function can be readily incorporated into our optimization procedure. Such an approach has been taken within the EGO framework by [7].

However, we have another use for derivatives. Covariance matrices over derivative observations are typically better conditioned than those over observations of the function itself—see Figure 3. Note that a derivative is only weakly correlated with the function very nearby, sidestepping conditioning problems due to close observations, but does provides useful information about the function at remoter locations around $r = \pm 1$. Relative to function observations, derivative observations are more weakly correlated with each other, meaning derivatives can be observed closer to each other at the same conditioning "cost." Similar conclusions are obtained under similarly shaped covariance functions, such as the rational quadratic and Matérn classes.

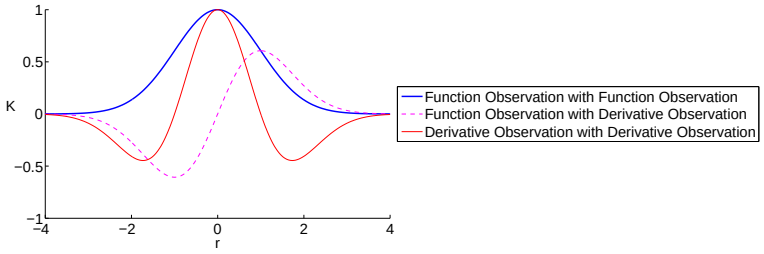

Fig. 3: The squared exponential covariance K between observations, of either the function or its derivative, separated by r.

Given this fact, our covariance matrix would be better conditioned if we had taken derivative observations in the place of function observations. As such, whenever we take an function evaluation (x_d, y_d) that is closer than δ to any existing observation (x_+, y_+), we simply do not incorporate it into our covariance matrix. In its place, we include a directional derivative observation that forces the slope at the midpoint of (x_d, y_d) and (x_+, y_+) to be that of the line that connects them. That is, our new derivative observation is $(\frac{1}{2}(x_+ + x_d), y_+ - y_d)$ along the direction $x_+ - x_d$. Given that x_+ and x_d were close, and that the GP retains knowledge of y_+, we can expect to lose very little information in making this linear approximation. Of course, we do not discard y_d altogether—it can be used for the purposes of computing future derivative observations, as illustrated in Figure 4. y_d also forms part of $\boldsymbol{y}_0$, used for the purposes of computing η, for

example. This use of derivative observations in the place of function observations sacrifices very little information, while giving rise to a much better conditioned algorithm.

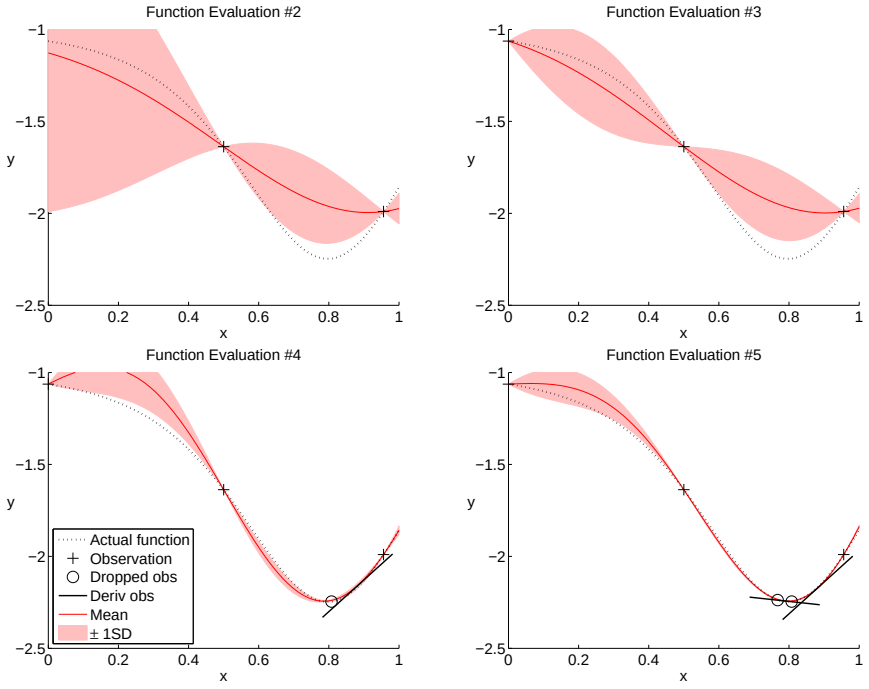

Fig. 4: An illustration of the use of derivative observations as a replacement for the excessively close function observations required in locating a minimum. Both function evaluation #4 and #5 are replaced by appropriately calculated derivative observations.

3.5 Confidence in Output

Note that, even if we do not sample exactly at a minimum, its location will often be evident in light of other nearby observations. Hence an obvious possibility is to allow the algorithm to report a suspected—rather than evaluated—minimum.

As such, we relax the constraint on our final returned value $x_M \in \boldsymbol{x}_0$. Instead, we merely require that $p(\, y_M \mid I_0 \,)$ has standard deviation equal to or less than some pre-specified threshold ε. Essentially, we quantify by ε exactly how confident we need to be in the returned value y_M. Of course, if ε is sufficiently small, usually only elements of $\boldsymbol{y}_0$ will be eligible to be returned by the algorithm—only points at which the function has actually been evaluated. This suggestion replaces η with $\eta_i' \triangleq \min_{x:C_i<\varepsilon^2} m_i$, now dependent on the hyperparameter values. Conditioning aside, this change will improve the performance of

our algorithm by obviating the necessity to expend a sample simply to return a function value we are already very confident in. As an example, in minimizing a function known to be quadratic, with the appropriate covariance function, any three observations away from the minimum will exactly determine where that minimum is. The use of η requires that we then evaluate the function at that point, whereas this new proposal allows us to end the procedure immediately.

3.6 Noisy Optimization

Many applications [8] require an effective means of noisy optimization. In such problems, rather than observing the objective function $y(x)$ directly, we instead observe $z(x)$, which differs from $y(x)$ in the addition of some noise component. For now, we restrict attention to Gaussian noise. In this case, we can easily apply our methods through a trivial [3] modification to the GP's covariance. We need simply to add the noise variance σ^2 to the diagonal of the covariance matrix over such observations[1], leading to

$$\begin{aligned} \boldsymbol{m}_\theta(\boldsymbol{z}_\star|I'_0) &= \boldsymbol{\mu}_\theta(\boldsymbol{x}_\star) + \mathbf{K}_\theta(\boldsymbol{x}_\star, \boldsymbol{x}_0)\big(\mathbf{K}_\theta(\boldsymbol{x}_0, \boldsymbol{x}_0) + \sigma^2\,\mathbf{E}_0\big)^{-1}(\boldsymbol{y}_0 - \boldsymbol{\mu}_\theta(\boldsymbol{x}_0)) \\ \boldsymbol{C}_\theta(\boldsymbol{z}_\star|I'_0) &= \\ &\quad \mathbf{K}_\theta(\boldsymbol{x}_\star, \boldsymbol{x}_\star) + \sigma^2\,\mathbf{E}_\star - \mathbf{K}_\theta(\boldsymbol{x}_\star, \boldsymbol{x}_0)\big(\mathbf{K}_\theta(\boldsymbol{x}_0, \boldsymbol{x}_0) + \sigma^2\,\mathbf{E}_0\big)^{-1}\mathbf{K}_\theta(\boldsymbol{x}_0, \boldsymbol{x}_\star)\,, \end{aligned} \tag{10}$$

where $\mathbf{E}$ is the identity matrix of appropriate size and I'_j is the conjunction of I and our noisy observations of the function, $\{(\boldsymbol{x}_{j'}, \boldsymbol{z}_{j'}) : j' = 1, \ldots, j\}$. Note also that $\boldsymbol{m}_\theta(\boldsymbol{y}_\star|I'_0) = \boldsymbol{m}_\theta(\boldsymbol{z}_\star|I'_0)$. We must also slightly modify our hyperparameter sample weights to $\boldsymbol{\rho}'$, to allow our likelihoods to incorporate the new covariance.

The noise variance is commonly unknown, in which case it can be treated just as any other hyperparameter. Henceforth, σ is treated as a member of θ, over which we now have samples S'. Of course, we should always exploit our prior knowledge about the problem at hand. If we suspect the objective function may be corrupted by noise, we should specify a prior over the noise variance and then marginalize using Bayesian Monte Carlo.

Of course, here the considerations of Section 3.5 take a more central importance. If the noise is considerable, we may in fact need to take several function evaluations at, or very near, the same point before we become sufficiently confident in the function's value there. Note that our ultimate loss remains the lowest function value known with sufficient confidence—there is no need for it to be subjected to any heuristic modifications [9]. As such, we have the loss function

$$\lambda'_i(z_j) \triangleq \min_{x\,:\,C_i(y|I'_j)<\varepsilon^2} m_i(y|I'_j)\,. \tag{11}$$

[1] Note that this modification has the fortunate side-effect of alleviating any conditioning problems.

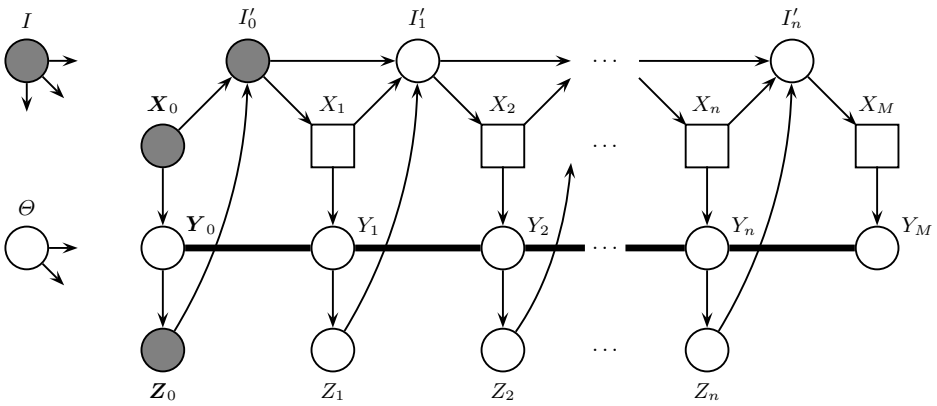

Fig. 5: A Bayesian network for the noisy optimization problem. Shaded nodes are known, and square nodes represent the results of a decision. All Y nodes are correlated with one another, the context I is correlated with all nodes and Θ is correlated with all Y and Z nodes.

and, from Figure 5, the new n-step lookahead expected loss

$$
\begin{aligned}
&\Lambda'_n(\, x_1 \,|\, I'_0\,) \\
&\triangleq \textstyle\sum_{i\in S'} \rho'_i \int \lambda'_i(z_n) \\
&\qquad \mathrm{N}\big(z_n; m_i(z_n|I'_{n-1}), C_i(z_n|I'_{n-1})\big)\delta\big(x_n - \operatorname{argmin}_{x_\star} \Lambda'_1(\, x_\star \,|\, I'_{n-1}\,)\big) \ldots \\
&\qquad\qquad \mathrm{N}\big(z_2; m_i(z_2|I'_1), C_i(z_2|I'_1)\big)\, \delta\big(x_2 - \operatorname{argmin}_{x_\star} \Lambda'_{n-1}(\, x_\star \,|\, I'_1\,)\big) \\
&\qquad\qquad\qquad \mathrm{N}\big(z_1; m_i(z_1|I'_0), C_i(z_1|I'_0)\big)\mathrm{d}z_1 \ldots \mathrm{d}z_n \mathrm{d}x_2 \ldots \mathrm{d}x_n \,. \qquad (12)
\end{aligned}
$$

As with those described in Section 3.2, these integrals must be approximated as weighted sums. Unfortunately, the analytic result that gave us (8) does not hold in the noisy case. Of course, if the noise is not too large relative to the scale of variation of the objective function, (8) will continue to be effective. Under this assumption, the 1-step lookahead expected loss is

$$
\Lambda'_1(\, x_1 \,|\, I'_0\,) \approx \sum_{i\in S'} \rho'_i\big(\eta'_i + (m'_i - \eta'_i)\, \Phi(\eta'_i; m'_i, C'_i) - C'_i\, \mathrm{N}(\eta'_i; m'_i, C'_i)\,\big)\,, \qquad (13)
$$

where we have abbreviated $m_i(z_1|I'_0)$ as m'_i, $C_i(z_1|I'_0)$ as C'_i and

$$
\eta'_i = \min_{x_1 : C'_i < \varepsilon^2} m'_i \,. \qquad (14)
$$

4 Results

We have compared the results of our method, Gaussian Process Global Optimization (GPGO), with several proposed alternatives on an extensive set of standard test functions for global optimization.

4.1 Testing Methodology

To compare the various optimization methods, we chose a diverse list of 14 standard test functions (see Table 1) for global optimization methods [10]. Each standard function additionally has a commonly used "standard" box-shaped test region containing the global minimum[2]. For noiseless testing, to reduce the possibility of an algorithm "stumbling" upon an optimal value rather than systematically finding it, for each problem we created ten associated subproblems by randomly translating the test region while guaranteeing that all minima remain within the box (except for multi-step lookahead problems, for which time restricted us to testing two each). This allows us to compare the average performance of each method on each problem. These translated subproblems also increase the difficulty of the test functions in several cases, because several of the functions are ill-behaved outside their standard test domain. A similar tactic was used to reduce sample bias during noisy testing; for each test problem and noise level, three different realizations of noise were used, and the performance for each method was determined from its aggregate performance.

Additionally, we tested the noiseless optimization routines on two- and three-dimensional functions generated from the GKLS random-function generator [11]. The default parameters were used, except the number of local minima was increased from ten to twenty; the first ten generated functions for each set of parameters formed the test set.

Because the objective functions we are considering are assumed to be very expensive to evaluate, we limit the number of allowed function evaluations. For noiseless problems, we use ten-times the dimension of the problem; for noisy problems, we use double this number. This restriction also applied to any initial function evaluations required to establish hyperparameters, if prescribed by the procedure under consideration. For very expensive functions, limiting the number of function evaluations is a natural stopping criterion.

We use the "gap" measure of performance to compare the methods [9]

$$G \triangleq \frac{y\left(x^{(\text{first})}\right) - y\left(x^{(\text{best})}\right)}{y\left(x^{(\text{first})}\right) - y^{(\text{opt})}}\,, \tag{15}$$

where y is the objective function, $y^{(\text{opt})}$ is the global optimum, and $x^{(\text{first})}$ and $x^{(\text{best})}$ are the first point evaluated and best point found at termination, respectively. For every method tested, the initial point $x^{(\text{first})}$ was chosen to be the medial point of the test region.

4.2 Implementation Details

The implementations used for the efficient global optimization (EGO) [2], the radial basis function (RBF) [12], and the dividing rectangles (DIRECT) [13] optimization algorithms were those in the TOMLAB/CGO toolbox for MATLAB,

[2] Note that our algorithm is not inherently limited to box-bounded problems; we can impose any arbitrary constraint by setting the expected loss to $+\infty$ over the region outside the desirable domain. The problem of constrained optimization is then passed onto the routine minimizing our expected loss.

version 6 [14]. The default settings were used for each, except that the number of function evaluations used in the initial samples (from which the methods derive their hyperparameters) was reduced by 25%. Otherwise, the methods would have used all of their available function evaluations just to build their model. Thereby we improve the performance of these algorithms by allowing them to intelligently—rather than systematically—choose where to place at least a portion of their function evaluations. The MATLAB implementations used for the noisy optimization routines Implicit Filtering (IF) [15], version 0.7, and Stable Noisy Optimization by Branch and Fit (SNOBFIT) [16], version 2.1, are freely available from the authors' web pages.

The optimization of the expected loss surface was performed using parallel subspace trust-region, interior-reflection Newton methods. Problematically for simple optimization procedures, the expected loss function is usually multimodal. Of course, even a local minimum of the expected loss is likely to represent a reasonably informative observation. Indeed, our results proved largely insensitive to changes in the method used for this optimization.

The covariance function used for modeling the response surface is given in (2). For functions known to have periodic components, the full covariance function was used; otherwise, only the non-periodic component was used by setting $h_B = 0$. We tested both the myopic one-step lookahead and the two-step lookahead versions of our algorithm, the latter only applied to the most demanding problems. For noisy optimization, we limited ourselves to the myopic one-step lookahead policy (13) with the non-periodic covariance. Such restrictions were made to accelerate testing (allowing more functions to be included in the test).

The input scales w_A and w_B and the output scales h_A and h_B were marginalized using sequential Bayesian Monte Carlo as described in Section 2. Suppose the test region is specified by the box $\prod_i [a_i, b_i]$, and let $m \triangleq \max_i(b_i - a_i)$ be the largest dimension of this box. The samples for these hyperparameters were chosen to be uniform in the log-space; specifically, the samples for $\log(w_A)$ and $\log(w_B)$ were chosen to be equally spaced in the interval $[\frac{m}{10} - \frac{3}{2}, \frac{m}{10} + \frac{3}{2}]$, and the samples for $\log h_A$ and $\log h_B$ were chosen to be equally spaced in the interval $[-2, 14]$. Five samples were used for each input scale parameter and nine samples were used for each output scale parameter.

For noisy optimization, the noise parameter σ was marginalized in a similar manner. The samples for σ were chosen to be uniform in the log-space in the interval $[\ln(0.01), 0]$. Five samples for the noise parameter were used. The threshold ϵ used was equal to 1 in all noisy tests.

4.3 Discussion

The results of testing are shown in Table 1. Our method performed (or tied with) the best out of the four algorithms tested for thirteen of the sixteen noiseless problems. Additionally, the grand mean performance of our method (even for the simplest model and least informative prior) surpasses the other methods by 16%. Clearly GPGO provides an improvement over other state-of-the-art solutions for global optimization of expensive functions.

Table 1: Mean measure of performance G for various global optimization techniques. The first table presents results over noiseless functions, the second table results for functions corrupted by Gaussian noise of various standard deviations. Numbers highlighted in bold are the highest for the relevant problem.

				GPGO 1-Step		GPGO 2-Step
	EGO	RBF	DIRECT	Non-Periodic	Periodic	Non-Periodic
Br	0.943	0.960	0.958	**0.980**	—	—
C6	0.962	0.962	0.940	0.890	—	**0.967**
G–P	0.783	0.815	**0.989**	0.804	—	**0.989**
H3	0.970	0.867	0.868	**0.980**	—	—
H6	0.837	0.701	0.689	**0.999**	—	—
Sh5	0.218	0.092	0.090	**0.485**	—	—
Sh7	0.159	0.102	0.099	**0.650**	—	—
Sh10	0.135	0.100	0.100	**0.591**	—	—
GK2	0.571	0.567	0.538	**0.643**	—	—
GK3	0.519	0.207	0.368	**0.532**	—	—
Shu	**0.492**	0.383	0.396	0.437	0.348	0.348
G2	0.979	**1.000**	0.981	**1.000**	**1.000**	—
G5	**1.000**	0.998	0.908	0.925	0.957	—
A2	0.347	0.703	0.675	0.606	0.612	**0.781**
A5	0.192	**0.381**	0.295	0.089	0.161	—
R	0.652	0.647	0.776	0.675	**0.933**	—
mean	0.610	0.593	0.604	**0.705**	—	—

	$\sigma = 0.1$			$\sigma = 0.2$			$\sigma = 0.5$		
	SNOBFIT	IF	GPGO	SNOBFIT	IF	GPGO	SNOBFIT	IF	GPGO
Br	0.980	**0.999**	0.996	0.983	**0.999**	0.997	0.979	**0.998**	0.993
C6	0.997	0.994	**0.998**	**0.999**	0.996	0.998	0.995	0.994	**0.997**
G–P	0.998	0.997	**1.000**	0.998	0.997	**1.000**	0.998	0.997	**1.000**
H3	0.956	0.954	**0.977**	0.972	0.921	**0.984**	0.881	0.563	**0.945**
H6	0.797	0.733	**0.936**	**0.864**	0.537	0.615	0.602	0.000	**0.864**
Sh5	0.066	0.001	**0.134**	0.128	0.002	**0.309**	0.001	0.002	**0.221**
Sh7	0.118	0.038	**0.158**	0.115	0.034	**0.278**	**0.052**	0.015	0.047
Sh10	0.133	0.284	**0.288**	0.130	0.100	**0.351**	0.032	0.002	**0.082**
Shu	0.667	0.417	**0.669**	0.488	0.452	**0.919**	**0.610**	0.419	0.590
G2	**1.000**	0.320	**1.000**	**1.000**	0.320	**1.000**	**1.000**	0.320	**1.000**
G5	**1.000**	0.195	**1.000**	**1.000**	0.195	**1.000**	**1.000**	0.195	**1.000**
A2	0.738	0.051	**0.768**	0.713	0.061	**0.808**	0.792	0.055	**0.902**
A5	**0.499**	0.053	0.293	**0.577**	0.051	0.369	**0.482**	0.043	0.269
R	0.000	**0.988**	0.000	0.000	**0.988**	0.000	0.030	**0.632**	0.000
mean	0.638	0.501	**0.658**	0.640	0.474	**0.696**	0.596	0.373	**0.636**

Function	Full Name	Test Region	Function	Full Name	Test Region
Br	Branin	$[-5, 10] \times [0, 15]$	Shu	Shubert	$[-10, 10]^2$
C6	Camelback 6	$[-5, 5]^2$	G2/5	Griewank 2/5	$[-600, 600]^D$
G–P	Goldstein–Price	$[-5, 5]^2$	A2/5	Ackley 2/5	$[-32.8, 32.8]^D$
H3/6	Hartman 3/6	$[0, 1]^D$	R	Rastrigrin	$[-5.12, 5.12]^2$
GK2/3	GKLS 2/3	$[-1, 1]^D$			
Sh5/7/10	Shekel 5/7/10	$[0, 10]^4$	(D is that indicated by the function name)		

Our performance can be partially explained by our proper handling of hyperparameters. By utilizing sequential Bayesian Monte Carlo methods we can intelligently select where to place function evaluations from the very beginning, learning more both about the model hyperparameters and the objective function with each new observation. This is in contrast with the naïve MLE or MAP approaches to hyperparameter management used by competing methods. Additionally, an improvement in performance can be seen when using a more informative prior covariance for periodic functions, illustrating the power of incorporating all available prior knowledge. We finally see an improvement when relaxing the strict myopic assumption. Combining our best results for each problem (using a periodic covariance or multi-step lookahead, as appropriate), our grand mean performance increases to 0.755, a 24% improvement over the closest tested competitor.

Our ability to exploit prior information is also effective in optimising functions corrupted by noise. As can be seen, our algorithm significantly outperformed tested competitors, exceeding or equalling their performance in twenty-nine of the forty-two problems. This is even more remarkable given the myopic approximation (13) used for testing. The use of a more sophisticated algorithm from (12) can be expected to even further improve our performance.

5 Conclusions and Future Work

We have proposed a novel method of optimization employing Gaussian processes, GPGO. It has clear benefits over older, similar work. Our formulation of optimization as a sequential decision problem allows us to state the exact Bayes optimal policy. This can either be approximated using a simple myopic approach, or, computation permitting, a more thorough multiple-step lookahead policy evaluated. We further use a computationally efficient sequential GP and employ Bayesian Monte Carlo to give a principled method of managing hyperparameters. This latter fact allows us to extract maximal information from all our function evaluations. Further benefits are found in our ability to exploit prior knowledge about the objective function, as in the cases that we suspect it may be periodic or corrupted by noise. The GP also allows the incorporation of any available derivative observations, giving a means to address conditioning issues.

A number of extensions to the work discussed here present themselves. Firstly, we may wish to tackle problems for which we have multiple modes of observation. Building upon our approach to noisy optimization, this could include functions for which we have a suite of variously costly, but also variously noisy, possible measurements. Applications are to be found in *multiple-fidelity* optimization and sensor-selection problems. Similarly, we may have to choose between an observation of the function itself or one of its derivatives. Essentially, these problems all reduce simply to the introduction of an additional input dimension to our problem, specifying which type of observation we take—a simple extension to the methods discussed above.

We might also wish to consider functions that are changing in time. Here we may have to select when to evaluate a function of time in order to maximize it. An example might be to determine the most profitable future time to make a costly financial trade. Alternatively, we might wish to optimize a dynamic function of which we can choose an observation only once every time step. Here we will be compelled to revisit the function's minima in order to confirm how they have changed with time. This offers applications to *concept-drift* problems.

References

1. Nocedal, J., Wright, S.J.: Numerical Optimization. Springer (July 2006)
2. Jones, D.R., Schonlau, M., Welch, W.J.: Efficient Global Optimization of Expensive Black-Box Functions. Journal of Global Optimization **13**(4) (1998) 455–492
3. Rasmussen, C.E., Williams, C.K.I.: Gaussian Processes for Machine Learning. MIT Press (2006)
4. Jones, D.R.: A Taxonomy of Global Optimization Methods Based on Response Surfaces. Journal of Global Optimization **21**(4) (2001) 345–383
5. Rasmussen, C.E., Ghahramani, Z.: Bayesian Monte Carlo. In Becker, S., Obermayer, K., eds.: Advances in Neural Information Processing Systems. Volume 15. MIT Press, Cambridge, MA (2003)
6. Osborne, M.A., Rogers, A., Ramchurn, S., Roberts, S.J., Jennings, N.R.: Towards real-time information processing of sensor network data using computationally efficient multi-output Gaussian processes. In: International Conference on Information Processing in Sensor Networks (IPSN 2008). (April 2008) 109–120
7. Leary, S., Bhaskar, A., Keane, A.: A Derivative Based Surrogate Model for Approximating and Optimizing the Output of an Expensive Computer Simulation. Journal of Global Optimization **30**(1) (2004) 39–58
8. Carter, R., Gablonsky, J., Patrick, A., Kelley, C., Eslinger, O.: Algorithms for Noisy Problems in Gas Transmission Pipeline Optimization. Optimization and Engineering **2**(2) (2001) 139–157
9. Huang, D., Allen, T.T., Notz, W.I., Zeng, N.: Global Optimization of Stochastic Black-Box Systems via Sequential Kriging Meta-Models. Journal of Global Optimization **34**(3) (2006) 441–466
10. Molga, M., Smutnicki, C.: Test functions for optimization needs (1995) `www.zsd.ict.pwr.wroc.pl/files/docs/functions.pdf`.
11. Gaviano, M., Kvasov, D.E., Lera, D., Sergeyev, Y.D.: Algorithm 829: Software for generation of classes of test functions with known local and global minima for global optimization. ACM Trans. Math. Softw. **29**(4) (2003) 469–480
12. Gutmann, H.M.: A Radial Basis Function Method for Global Optimization. Journal of Global Optimization **19**(3) (2001) 201–227
13. Jones, D.R., Perttunen, C.D., Stuckman, B.E.: Lipschitzian optimization without the Lipschitz constant. Journal of Optimization Theory and Applications **79**(1) (1993) 157–181
14. Holmström, K.: New Optimization Algorithms and Software. Theory of Stochastic Processes **1**(2) (1999) 55–63
15. Kelley, C.: Users Guide for imfil Version 0.7 (2008) `http://www4.ncsu.edu/~ctk/MATLAB_IMFIL07/manual.pdf`.
16. Huyer, W., Neumaier, A.: SNOBFIT–Stable Noisy Optimization by Branch and Fit. ACM Transactions on Mathematical Software **35** (2008)