Surrogate-Based Global Optimization with Parallel Function Evaluation

Raphael T. Haftka
Distinguished Professor, haftka@ufl.edu
Felipe A. C. Viana
Research Assistant, fccheury@ufl.edu
Department of Mechanical and Aerospace Engineering,
University of Florida, Gainesville, FL, 32611-6250, USA

Abstract: Surrogate-based techniques have become popular because they reduce the computational burden associated design optimization of complex engineering systems. In each optimization cycle, a surrogate is fitted to a number of previously analyzed designs. A candidate solution is then found from optimization using the surrogate. Finally the candidate solution is analyzed (running expensive simulation). If it appears that further improvements can be achieved, we update the surrogate with this new sampled point (and possibly zoom in on regions of interest) and conduct another optimization cycle. Algorithms that use the surrogate uncertainty estimator to guide the selection of the next sampling candidate are readily available, e.g., the efficient global optimization (EGO) algorithm. However, adding one point at a time may not be efficient in applications where one can run simulations in parallel, and simulations might abort for some design configurations (the main concern is shifted to wall-clock time rather than the total number of simulations). Additionally, the need for uncertainty estimates limits adaptive sampling to surrogates such as kriging and polynomial response surface that are normally implemented with uncertainty estimates. We propose the multiple surrogate efficient global optimization (MSEGO) algorithm, which adds several points per optimization cycle with the help of multiple surrogates. We import uncertainty estimates from one surrogate to another to allow use of surrogates that do not provide them. The approach is tested on three analytic examples for nine basic surrogates including kriging, radial basis neural networks, linear Shepard, and support vector regression. We found that (i) our MSEGO works well with the imported uncertainty estimates, and (ii) MSEGO delivered better results in a fraction of the optimization cycles needed by EGO.

1. Introduction: Savings in computational time popularize surrogate-based optimization in the engineering design community [1]-[9]. Instead of using expensive simulation codes directly, the optimization begins by fitting a surrogate (also known as metamodel or response surface approximation) to data from a number of simulations. In each cycle, the goal is to obtain a set of designs that are candidate solutions of the optimization. Besides the prediction offered by surrogates, modern surrogate-based optimization strategies also use error estimates. For example, the efficient global optimization (EGO [10]) and the sequential kriging optimization (SKO [11]) algorithms use the kriging [12]-[14] prediction and prediction variance to seek the point of maximum expected improvement as the next simulation for the optimization process.

Traditional implementations of EGO-like algorithms add a single simulation per cycle. However, opportunities for parallel computing and the human effort associated with setting up complex simulations drive complex applications towards running as many simulations as possible per cycle. The fragility of many complex simulations (i.e., they can abort) also encourages cycles with large number of simulations, which are less sensitive to a few failed simulations. In addition, in many engineering applications, it may take weeks to complete simulations, and only very few cycles are undertaken. Hence, there is a strong incentive to overcome the limitation of adding a single data point per cycle. Research in this direction has extended the expected improvement to include multiple points [15]-[18]. However, the optimization of the design of experiments that will maximize the multiple-point expected improvement is computationally challenging [17], [18]. The strategy we propose is motivated by reported success in the use of multiple surrogates for optimization [19], [20]. These approaches usually involve: (i) fitting multiple surrogates (e.g, kriging, polynomial response surface [21], [22], radial basis neural network [23], [24], Shepard interpolation [25], [26], and support vector regression [27], [28]) and picking one based on a figure of merit (e.g., estimators of the root mean square error [29], [30]), and (ii) performing optimization multiple times with multiple surrogates [31], [32].

We propose a strategy for adding several points per optimization cycle based on the simultaneous use of multiple surrogates. As suggested by Viana and Haftka.
We import uncertainty estimates from one surrogate to another to enable running EGO with the surrogates that do not furnish error estimates (such as support vector regression models). Appendix A summarizes the scheme for importing uncertainty estimates. Then, we use the pool of surrogates to provide multiple points per cycle for the EGO algorithm. We do not aim to outperform kriging, instead we expect to reduce the number of cycles for convergence and yet deliver comparable results.

The remainder of the paper is organized as follows. Section 2 presents the background necessary for the development of the proposed approach. Section 3 introduces our method for running EGO with multiple surrogates. Section 4 presents the results of numerical experiments and some discussion. Section 5 closes the paper with concluding remarks.

2. Background

2.1 Sampling Driven by Optimization: We just give an overview of the efficient global optimization (EGO) algorithm by Jones et al. [10]. EGO starts by constructing a kriging surrogate interpolating the initial set of data points. Kriging models the unknown objective function $y(x)$ as a realization of a Gaussian process $Y(x)$. Kriging estimates as a combination of basis functions $f_i(x)$ (e.g., a polynomial basis) and departures $z(x)$ (representing low and high frequency variation components, respectively) by

$$
y(x) = \sum_{i=1}^{m} \beta_i f_i(x) + z(x), \tag{1}\n$$

where $z(x)$ satisfies $z(x_k) = y(x_k) - \sum_{i=1}^{m} \beta_i f_i(x_k)$ for all sample points $(x_k)$ and is assumed to be a realization of a stochastic process $Z(x)$ with mean zero, process variance $\sigma^2$, and spatial covariance function given by

$$
\text{cov}(Z(x_i), Z(x_j)) = \sigma^2 R(x_i, x_j), \tag{2}\n$$

$$
\sigma^2 = \frac{1}{p} (y - Xb)^T R^{-1} (y - Xb), \tag{3}\n$$

where $R(x_i, x_j)$ is the correlation between $Z(x_i)$ and $Z(x_j)$, $y$ is the value of the actual responses at the sampled points, $X$ is the Gramian design matrix constructed using the basis functions at the sampled points, $R$ is the matrix of correlations $R(x_i, x_j)$ among function values at sample points, and $b$ is an approximation of the vector of coefficients $\beta_i$ of Eq. (1). See [12] to [14] for details on the estimation of the kriging parameters.

We can estimate the uncertainty in $\hat{y}(x)$ using the kriging prediction variance (also known as mean squared error of the predictor)

$$
s^2(x) = \sigma^2 \left(1 + u^T X^T R^{-1} X \right)^{-1} u - r^T R^{-1} r \tag{4}\n$$

where $u = X^T R^{-1} r - f$, $r$ is the vector of correlations between function values at the point $x$ and the design points, $f$ is the vector of basis functions at the point $x$. For more details about the kriging prediction variance, see [12] to [14].

Figure 1 illustrates both the prediction and the error estimates of kriging; since the kriging model used here is an interpolator, the error vanishes at data points.

![Figure 1. Kriging model $\hat{y}_{KRG}(x)$ of the function $y(x) = (6x - 2)^2 \sin(12x - 4)$ sampled at $x = [0 \ 0.5 \ 0.68 \ 1]^T$. The uncertainty (whose amplitude is $1.75 \times s(x)$) associated with $\hat{y}_{KRG}(x)$ is shown in gray.](image)

After constructing the kriging model, the algorithm iteratively adds points to the data set in an effort to improve upon the present best sample $y_{PBS}$. The improvement at a point $x$ is

$$
I(x) = \max(y_{PBS} - Y(x), 0), \tag{5}\n$$
which is a random variable because \( Y(x) \) is a random variable (recollect that kriging models the response \( y(x) \) as a realization of a Gaussian process \( Y(x) \)). The formal derivation of an expression for the expected improvement \( E[I(x)] \) can be found in [10]. Here we just show it as it is
\[
E[I(x)] = s(x)[\nu \Phi(\nu) + \phi(\nu)],
\]
\[
u = \left[ y_{PBS} - \hat{y}(x) \right] / s(x),
\]
(6)
where \( \Phi(\cdot) \) and \( \phi(\cdot) \) are the cumulative density function and probability density function of a normal distribution, and \( s(x) \) is the prediction standard deviation (here estimated as the square root of the prediction variance).

After adding the new point to the existing data set, the kriging model is updated (usually without the costly update of the correlation parameters). Figure 2 illustrates two cycles of the efficient global optimization (EGO) algorithm. Figure 2-(a) shows the initial kriging model and the corresponding expected improvement. The maximization of \( E[I(x)] \) adds \( x = 0.19 \) to the data set. This cycle is driven towards exploration of the design space. In the next cycle, EGO uses the updated kriging model shown in Figure 2-(b). This time, the maximization of \( E[I(x)] \) adds \( x = 0.74 \) to the data set. This is an exploitation cycle.

EGO iterates until the stopping criterion is met. Due to high computational cost of actual simulations, it is common to use the maximum number of function evaluations as the stopping criterion. Another alternative is to set a target value for the expected improvement. The original EGO, proposed in [10], is limited to surrogates such as kriging that have uncertainty estimates. In the next section we describe how to use surrogates without such estimates in EGO.

2.2 Generating Multiple Surrogates: Most practitioners in the optimization community are familiar at least with the traditional polynomial response surface, some with more sophisticated models such as kriging, neural networks, or support vector regression, and few with the use of weighted average surrogates [35]-[37]. The diversity of surrogate models might be explained by three basic components [20]:

1. Statistical modeling: for example, response surface techniques frequently assume that the data is noisy and the obtained model is exact. On the other hand, kriging usually assumes that the data is exact and is a realization of a Gaussian process.
2. Basis functions: response surfaces frequently use monomials. Support vector regression specifies the basis in terms of a kernel (many different functions can be used).
3. Loss function: the minimization of the mean square error is the most popular criteria for fitting the surrogate. Nevertheless, there are alternative measures such as the average absolute error (i.e., the L1 norm).

It is also possible to create different instances of the same surrogate technique. For example, we could create polynomials with different choice of monomials, kriging models with different correlation functions (see [38] for details), and support vector regression models.
with different kernel and loss functions (see [39] for details).

If multiple surrogates can be generated, we might want to check their accuracy. We do that with the help of the root mean square error

$$RMS_e = \sqrt{\frac{1}{V} \int_D \left( \hat{y}(x) - y(x) \right)^2 \, dx},$$

where $\hat{y}(x)$ is the surrogate model of the response $y(x)$, and $V$ is the volume of the design domain $D$.

We might estimate $RMS_e$ via Monte Carlo integration using test points. This is a precise assessment of accuracy, but the cost might be prohibitive in engineering applications. Cross validation (assessment based on data points) is an attractive alternative to that because it does not depend on the statistical assumptions of a particular surrogate technique and it does not require extra expensive simulations. Nevertheless, cross validation should be used with caution, since the literature has reported problems such as bias in error estimation [40], [41].

A cross-validation error is the error at a data point when the surrogate is fitted to a subset of the data points not including this point. When the surrogate is fitted to all the other $p - 1$ points, the process has to be repeated $p$ times (leave-one-out strategy) to obtain the vector of cross-validation errors, $e_{XV}$. Alternatively, the $k$-fold strategy can also be used for computation of the $e_{XV}$ vector. According to the classical $k$-fold strategy [29], after dividing the available data ($p$ points) into $p/k$ clusters, each fold is constructed using a point randomly selected (without replacement) from each of the clusters. Of the $k$ folds, a single fold is retained as the validation data for testing the model, and the remaining $k-1$ folds are used as training data. The cross-validation process is then repeated $k$ times with each of the $k$ folds used exactly once as validation data. Figure 3 illustrates computation of the cross-validation errors for a kriging surrogate.

![Figure 3: Cross-validation error at the second point of the DOE, $e_{XV2}$, exemplified by fitting a kriging model (KRG) to $p = 6$ data points.](image)

The square root of the PRESS value ($PRESS$ stands for prediction sum of squares) is the estimator of the $e_{RMS}$

$$PRESS_{RMS} = \sqrt{\frac{1}{p} e_{XV}^T e_{XV}}.$$  (8)

Figure 4 illustrates different surrogate fitted to the same set of points (different instances of kriging and support vector regression). First, we observe that different instances of the same surrogate might have very different accuracy levels (see the support vector regression case). Additionally, although $PRESS_{RMS}$ might not be extremely accurate estimator of $e_{RMS}$, it is good enough for surrogate selection (for a deeper discussion on cross validation, see [30]).

In optimization, it seems advantageous to use multiple surrogates. After all, one surrogate may be more accurate in one region of design space while another surrogate may be more accurate in a different region. The hope is that a set of surrogates would allow exploration of different portions of the design space by pointing to different candidate solutions of the optimization problem. For instance, Samad et al. [19] used polynomial response surface, kriging, radial basis neural network, and weighted average surrogate in a compressor blade shape optimization of the NASA rotor 37. It was found that the most accurate surrogate did not always lead to the best design. This demonstrated that using multiple surrogates can improve the robustness of the optimization at a minimal computational cost (insurance policy against poorly fitted models).
Proceedings of 2011 NSF Engineering Research and Innovation Conference, Atlanta, Georgia

Grant #085643

(a) Kriging surrogates with different correlation functions.

(b) Support vector regression models with different kernel functions.

Figure 4. Different surrogates models fitted to five data points of the function $y(x) = (6x - 2)^2 \times \sin(2 \times (6x - 2))$.

3. Efficient Global Optimization with Multiple Surrogates: We assume that multiple surrogates are available – with native or imported error estimates. Appendix A summarizes the scheme for importing uncertainty estimates for a surrogate from another surrogate. We propose running the efficient global optimization (EGO) algorithm with multiple surrogates simultaneously. In each optimization cycle, we maximize the expected improvement (6) of each surrogate of the set. We call it the multiple surrogate efficient global optimization (MSEGO) algorithm. We hope that the multiple surrogates potentially suggest multiple points that will reduce the number of cycles EGO needs for convergence. In terms of wall clock time, MSEGO is particularly advantageous if running one or multiple simulations takes approximately the same time (true for parallel computation of the simulations).

(a) Maximizing $E[I(x)]$ of the kriging model.

(b) Maximizing $E[I(x)]$ of the support vector regression model.

(c) Updating kriging and support vector regression models.

Figure 5. Global search with MSEGO and two surrogates.

The function $y(x) = (6x - 2)^2 \times \sin(12x - 4)$ is initially sampled at $x = [0, 0.5, 0.68, 1]$. Figure 5-(a) and Figure 5-(b) show how we select two points for the next cycle. Figure 5-(c) illustrates the updated models after $x = 0.19$ and $x = 0.77$ are added to the set.
For simplicity, we illustrate the approach using only two surrogates, kriging and support vector regression (that imports the prediction variance from the kriging model). MSEGO iteratively adds two points to the data set that come from the individual maximization of the expected improvement of both surrogates. After adding the new points to the existing data set, both surrogates are updated. Figure 5 illustrates one cycle of the MSEGO algorithm running with these two surrogates. Figure 5-(a) shows the initial kriging model and the corresponding expected improvement. The maximization of the kriging $E[I(x)]$ suggests adding $x = 0.19$ to the data set. Figure 5-(b) shows the initial support vector regression model and the corresponding expected improvement. Here, the expected improvement suggests adding $x = 0.77$. We add both points and in the next cycle, our algorithm uses the updated models shown in Figure 5-(c). Here the use of two surrogates favors exploration of the design space since the suggested points are far apart.

If the two points are close, they may instead accelerate the local search as illustrated in Figure 6. Figure 6-(a) shows that by maximizing the kriging expected improvement, we would add $x = 0.69$ to the data set. Figure 6-(b) shows that the support vector regression suggests adding $x = 0.81$ to the data set. After including both points we obtain the updated models shown in Figure 6-(c).

In terms of implementation, one might want to avoid points that are closer together than a given threshold for surrogates such as kriging that suffer from ill conditioning in that case.

4. Numerical Experiments

4.1 Test Set: Table 1 details the different surrogates used during this investigation. The DACE toolbox of Lophaven et al. [38], the native neural networks MATLAB toolbox [43], and the code developed by Gunn [39] were used for kriging, the radial basis neural network, and support vector regression algorithms, respectively. The SURROGATES toolbox of Viana [44] was used to run the Shepard (adapted from SHEPPACK [45]) and the MSEGO algorithms.

- The interested reader can certainly find other packages (e.g., those available at http://www.kernel-machines.org, http://www.support-vectormachines.org, http://www.sumo.intec.ugent.be, the free companion code of [46], and the DiceKriging and DiceOptim packages [47]) (retrieved 2010).
Table 1: Set of surrogates used in the study of EGO assisted by multiple surrogates.

<table>
<thead>
<tr>
<th>Surrogates</th>
<th>Details</th>
<th>Provide uncertainty estimate?</th>
</tr>
</thead>
<tbody>
<tr>
<td>krg</td>
<td>Kriging model: Constant trend function and Gaussian correlation. $\theta_{li} = \left( \frac{1}{n_v} \right)$, and $10^{-3} \leq \theta_i \leq 2 \times \theta_{li}$, $i = 1, 2, \ldots, n_v$ were used.</td>
<td>YES</td>
</tr>
<tr>
<td>rbnn</td>
<td>Radial basis neural network: Goal $y = \text{Spread} = \frac{1}{\sqrt[3]{p}}$.</td>
<td>NO</td>
</tr>
<tr>
<td>shep</td>
<td>Linear Shepard model: Subroutine LSHEP from SHEPPACK [45].</td>
<td>NO</td>
</tr>
<tr>
<td>svr-grbf-e-full</td>
<td>Support vector regression: “grbf” and “poly” indicate the kernel functions (Gaussian and second order polynomial respectively).</td>
<td>NO</td>
</tr>
<tr>
<td>svr-grbf-e-short</td>
<td>“e” and “q” indicate the loss functions (“e” for $\varepsilon$-insensitive and “q” for quadratic).</td>
<td></td>
</tr>
<tr>
<td>svr-grbf-q</td>
<td>“full” and “short” refer to different values for the regularization parameter, $C$, and for the insensitivity, $\varepsilon$; “full” adopts $C = \infty$ and $\varepsilon = 1 \times 10^{-4}$, while “short” uses $\varepsilon = \sigma_y / \sqrt{p}$ and $C = 100 \max \left[ \bar{y} + 3 \sigma_y,</td>
<td>\bar{y} - 3 \sigma_y</td>
</tr>
<tr>
<td>svr-poly-e-full</td>
<td>“full” and “short” refer to different values for the regularization parameter, $C$, and for the insensitivity, $\varepsilon$; “full” adopts $C = \infty$ and $\varepsilon = 1 \times 10^{-4}$, while “short” uses $\varepsilon = \sigma_y / \sqrt{p}$ and $C = 100 \max \left[ \bar{y} + 3 \sigma_y,</td>
<td>\bar{y} - 3 \sigma_y</td>
</tr>
<tr>
<td>svr-poly-e-short</td>
<td>“full” and “short” refer to different values for the regularization parameter, $C$, and for the insensitivity, $\varepsilon$; “full” adopts $C = \infty$ and $\varepsilon = 1 \times 10^{-4}$, while “short” uses $\varepsilon = \sigma_y / \sqrt{p}$ and $C = 100 \max \left[ \bar{y} + 3 \sigma_y,</td>
<td>\bar{y} - 3 \sigma_y</td>
</tr>
</tbody>
</table>

As test problems, we employed the following two analytical benchmark problems:

- Sasena function [48] (two variables, see Figure 7)

\[
y(x) = 2 + 0.01 \left( x_2 - x_2^2 \right)^2 + \left( 1 - x_1 \right)^2 \\
+ 2 \left( 2 - x_2 \right)^2 + 7 \sin \left( 0.5 x_1 \right) \sin \left( 0.7 x_1 x_2 \right),
\]

\[
0 \leq x_1 \leq 5, \ 0 \leq x_2 \leq 5;
\]

- Hartman functions [49] (three and six variables, respectively)

\[
y(x) = -\sum_{i=1}^{4} a_i \exp \left( -\sum_{j=1}^{n_{dv}} B_{ij} (x_j - D_{ij})^2 \right),
\]

\[
a = \begin{bmatrix} 1.0 & 1.2 & 3.0 & 3.2 \end{bmatrix},
\]

\[
0 \leq x_j \leq 1, j = 1, 2, \ldots, n_{dv}.
\]

We use two instances: Hartman3 with three variables and Hartman6 with six variables, with their parameters shown in Table 2.

Table 2: Parameters used in Hartman functions.

<table>
<thead>
<tr>
<th>Function</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hartman3</td>
<td>\begin{bmatrix} 3.0 &amp; 10.0 &amp; 30.0 \end{bmatrix} \begin{bmatrix} 0.1 &amp; 10.0 &amp; 35.0 \end{bmatrix} \begin{bmatrix} 3.0 &amp; 10.0 &amp; 30.0 \end{bmatrix} \begin{bmatrix} 0.1 &amp; 10.0 &amp; 35.0 \end{bmatrix}</td>
</tr>
<tr>
<td>Hartman6</td>
<td>\begin{bmatrix} 10.0 &amp; 3.0 &amp; 17.0 &amp; 3.5 &amp; 1.7 &amp; 8.0 \end{bmatrix} \begin{bmatrix} 0.05 &amp; 10.0 &amp; 17.0 &amp; 0.1 &amp; 8.0 &amp; 14.0 \end{bmatrix} \begin{bmatrix} 3.0 &amp; 3.5 &amp; 1.7 &amp; 10.0 &amp; 17.0 &amp; 8.0 \end{bmatrix} \begin{bmatrix} 17.0 &amp; 8.0 &amp; 0.05 &amp; 10.0 &amp; 0.1 &amp; 14.0 \end{bmatrix}</td>
</tr>
</tbody>
</table>

To average out the influence of the initial data set, we repeat the experiments with 100 different Latin hypercube designs [50], [51]. The experimental designs are created by the MATLAB Latin hypercube function.
lhsdesign, using the “maxmin” option with 1,000 iterations.

We start sampling Sasena, Hartman3, and Hartman6 with 12, 20, and 56 points, respectively. Then, we let EGO run for six, ten, and fourteen cycles for Sasena, Hartman3, and Hartman6 functions, respectively. We run our algorithm with two, four, and nine surrogates with one of them being kriging. Given the experimental design, we select the surrogates that will assist kriging based on PRESS_{RMS}. We pair kriging with the surrogates with smallest PRESS_{RMS} in the set. In each cycle, we add at most the same number of points as the number of surrogates (that is two, four, or nine points per cycle but avoiding repeated points) until the maximum number of optimization cycles is reached (i.e., six and ten for the Sasena and the Hartman3 functions, respectively). Full details are given in Table 3.

Table 3: Data for EGO for the test problems. Surrogates with small PRESS_{RMS} are chosen to pair with kriging

<table>
<thead>
<tr>
<th>Setup</th>
<th>Sasena</th>
<th>Hartman3</th>
<th>Hartman6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Points in the initial experimental design</td>
<td>12</td>
<td>20</td>
<td>56</td>
</tr>
<tr>
<td>Maximum number of optimization cycles</td>
<td>6</td>
<td>10</td>
<td>14</td>
</tr>
<tr>
<td>Number of function evaluations running EGO with kriging (1 point per cycle)</td>
<td>6</td>
<td>10</td>
<td>14</td>
</tr>
<tr>
<td>Maximum number of function evaluations running EGO with 2 surrogates (at most 2 points per cycle)</td>
<td>12</td>
<td>20</td>
<td>28</td>
</tr>
<tr>
<td>Maximum number of function evaluations running EGO with 4 surrogates (at most 4 points per cycle)</td>
<td>24</td>
<td>40</td>
<td>56</td>
</tr>
<tr>
<td>Maximum number of function evaluations running EGO with 9 surrogates (at most 9 points per cycle)</td>
<td>54</td>
<td>90</td>
<td>126</td>
</tr>
</tbody>
</table>

4.2 Performance Measure: Given the initial best solution y_{IBS} and the solution y_{optm} found in the optimization task, we can compute the actual improvement \( I \) which measures the performance of the optimization algorithm

\[
I = y_{IBS} - y_{optm} .
\]  

The actual improvement \( I \) captures how well the optimization performed. \( I = 0 \) if there is no improvement, and \( I > 0 \) otherwise (\( I \) can be large in case of substantial improvement). A second measure is the ratio between the actual improvement \( I \) and the maximum possible improvement

\[
r_I = \frac{I}{I_{max}} , \quad I_{max} = y_{IBS} - y^* ,
\]

where \( y^* \) is the value of the function at the global optimum point. \( r_I \) ranges from zero (no improvement) to one (\( y^* = y_{optm} \)).

4.3. Results and Discussion: First, we checked how well cross validation selects the surrogates to be paired with kriging. Figure 8 gives box plots of both PRESS_{RMS} and \( e_{RMS} \) for all surrogates for all test problems (Appendix B describes box plots). For all test functions there is at least one surrogate that is as good as kriging in terms of \( e_{RMS} \). For the Sasena function, Figure 8-(a) shows that the support vector regression models with the polynomial kernel (variations of “svr-poly”) may outperform kriging in terms of \( e_{RMS} \). For Hartman3, Figure 8-(b) illustrates that kriging is comparable to the radial basis neural network (“rbnn”). For Hartman6, Figure 8-(c) shows that most of the surrogates are equally good, except for “svr-grbf-e-full” and “shep” that are just slightly less accurate than the other surrogates. The selection of the surrogates based on PRESS_{RMS} is almost the same as that based on \( e_{RMS} \). Table 4 shows how the surrogates rank according to overall performance. The match for Hartman3 and Hartman6 is very good, but not so good for Sasena. The rank according to PRESS_{RMS} and \( e_{RMS} \) is the same for Hartman6. Overall, most of the time the surrogates chosen to assist kriging will be the most accurate ones.
Table 4: Ranking of the surrogates according to median values (over 100 experimental designs) of \( \text{PRESS}_{\text{RMS}} \) and \( \epsilon_{\text{RMS}} \). \( \epsilon_{\text{RMS}} \cdot \text{PRESS}_{\text{RMS}} \) satisfactorily ranks the surrogates.

<table>
<thead>
<tr>
<th>Rank</th>
<th>Sasena, 12 points</th>
<th>Hartman3, 20 points</th>
<th>Hartman6, 56 points</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>svr-poly-q</td>
<td>svr-poly-e-full</td>
<td>rbnn</td>
</tr>
<tr>
<td>2</td>
<td>svr-poly-e-full</td>
<td>svr-poly-q</td>
<td>krg</td>
</tr>
<tr>
<td>3</td>
<td>svr-poly-e-short</td>
<td>svr-poly-e-short</td>
<td>svr-poly-e-short</td>
</tr>
<tr>
<td>4</td>
<td>svr-grbf-q</td>
<td>svr-grbf-q</td>
<td>svr-grbf-q</td>
</tr>
<tr>
<td>5</td>
<td>svr-grbf-e-short</td>
<td>svr-grbf-e-short</td>
<td>svr-grbf-e-short</td>
</tr>
<tr>
<td>6</td>
<td>krg</td>
<td>krg</td>
<td>krg</td>
</tr>
<tr>
<td>7</td>
<td>shep</td>
<td>shep</td>
<td>shep</td>
</tr>
<tr>
<td>8</td>
<td>rbnn</td>
<td>svr-poly-e-full</td>
<td>svr-poly-e-full</td>
</tr>
<tr>
<td>9</td>
<td>svr-grbf-e-full</td>
<td>svr-grbf-e-full</td>
<td>svr-grbf-e-full</td>
</tr>
</tbody>
</table>

Figure 8: Box plots of \( \text{PRESS}_{\text{RMS}} \) and \( \epsilon_{\text{RMS}} \) of surrogates for the test problems (over 100 experimental designs). Data obtained with the initial experimental designs (12, 20, and 56 points for Sasena, Hartman3, and Hartman6, respectively). In terms of prediction, other surrogates might be just as good as kriging. Appendix B explains box plots.

Figure 9 shows the median of the relative improvement ratio, defined in Eq. (12), out of the 100 experimental designs for the traditional efficient global optimization (EGO) algorithm (running only with kriging) and MSEGO (EGO assisted by multiple surrogates). While it is obvious that using multiple simulations per cycle should accelerate convergence, the improvement is quite high, especially for the Hartman6 function. There, using four surrogates for two cycles provides the same improvement as using kriging alone for 14 cycles. For all cases, the more points added per cycle (i.e., more surrogates), the faster the convergence. Because of that, the remainder of the paper will focus on the ensemble of nine surrogates.

Figure 9: Median (over 100 experimental designs) of the relative improvement ratio (12) as a function of the number of cycles. More points per cycle (more surrogates) speeds up performance.
Figure 10 shows box plots of the relative improvement ratio, defined in Eq. (12), out of the 100 experimental designs for the traditional EGO (running only with kriging) and MSEGO (EGO assisted by nine surrogates) iteration by iteration for Hartman3. A benefit from the diversity is the reduced dispersion of the results. Due to parallel computation, in terms of wall clock time, our approach takes only a fraction of the time a traditional EGO implementation would need.

Using multiple surrogates is mostly attractive when the main concern is the number of cycles rather than the total number of simulations. However, it is interesting to look at what penalty is paid for accelerated time convergence by having to run more simulations. For that purpose, compare the algorithms for a fixed number of total simulations rather than a fixed number of cycles. Figure 11 shows the median over 100 experimental designs of the relative improvement ratio, defined in Eq. (12), with respect to number of function evaluations for EGO and MSEGO. For Sasena there is a substantial penalty in the number of function evaluations and for Hartman3 there is a significant penalty early on, but better final convergence. For Hartman6 the multiple surrogate approach has a distinct advantage.

Overall, we believe that the difference in performance between traditional EGO and MSEGO is due to a combination of factors. One reason is that since the traditional EGO uses only kriging, it is sensitive to experimental designs for which the kriging model is a poor surrogate. On the other hand, MSEGO would have other surrogates to balance that. Another reason is that diversity of surrogates tends to improve both exploration and exploitation. Figure 12 shows two intersite distance measures of the final data set (we computed the distance between all possible pairs). The median intersite distance (left-hand plots) illustrates how much exploration there is. In all test problems, MSEGO tends to spread the points over the design space more than kriging alone. On the other hand, MSEGO does not prevent points from clustering. This makes the minimum intersite distance (right-hand plots of Figure 12 for Sasena and Hartman3) smaller compared to traditional EGO (although this tendency is reversed for Hartman6). Since we are aiming to reduce the number of cycles needed to achieve a certain relative improvement, we see that the benefits of faster exploration counterbalance the effects of clustering (on top of that, Figure 6 illustrates that having points close to each other may not necessarily be bad).
Proceedings of 2011 NSF Engineering Research and Innovation Conference, Atlanta, Georgia

Grant #0856431

6. Concluding Remarks: We proposed the multiple surrogate efficient global optimization (MSEGO) algorithm, which uses multiple surrogates to add multiple design sites at each cycle of the efficient global optimization (EGO) algorithm. The approach is based on importing uncertainty estimates to furnish such structures for the surrogates that lack them in their original implementation. MSEGO is a cheap alternative to adding multiple points in each EGO cycle based on single kriging model (computationally very difficult).

Three algebraic examples and nine surrogates were used to study how well importing uncertainty estimates works and to compare the traditional implementation of EGO (running with kriging alone) with MSEGO (EGO assisted by a set of surrogates). For these examples we found that

- the imported uncertainty structure allowed non-kriging surrogates to be used in the EGO algorithm;
- MSEGO reduced substantially the number of cycles required for convergence;
- the penalty in terms of total number of function evaluations was substantial for one of the three examples, but for another MSEGO provided a substantial reduction in the number of function evaluations and a very large reduction in the number of cycles.

8. Acknowledgements: This work was supported by the National Science Foundation grant CMMI-0856431. We also would like to thank Mr. David Easterling and Mr. Nick Radcliffe for their help in the implementation of the Shepard interpolator.

Appendices

A. Importing Error Estimates from Another Surrogate: Viana and Haftka [33] developed the heuristic rationale behind the importation of uncertainty estimates. Here, we briefly illustrate the mechanism with a simple example. Suppose that a kriging (KRG) and a support vector regression (SVR) models are built from five points, as illustrated in Figure 13-(a). The KRG model has a Gaussian correlation function and a constant for the low frequency component (trend function); the SVR model uses a Gaussian kernel function with $\varepsilon = 0$ for the $\varepsilon$-insensitive loss function. As a result, both surrogates are interpolators. Viana and Haftka [33] proposed combining the predictor of a model with the uncertainty estimate of another model. By comparing Figure 13-(b) and (c), we can see that kriging offers a reasonable error estimate for the support vector regression (in this example).

Figure 13. Importation of uncertainty estimates. Figure 13-(a) shows a kriging (KRG) and a support vector regression (SVR) models (both surrogates act as interpolators). Figure 13-(b) shows the KRG model and the square root of its prediction variance (in gray). Figure 13-(c) shows the SVR model with the square root of the prediction variance borrowed from KRG (in gray). Although not based on the SVR statistical assumptions, the KRG uncertainty estimate offers a reasonable replacement.
B. Box Plots: In a box plot, the box is defined by lines at the lower quartile (25%), median (50%), and upper quartile (75%) values. Lines extend from each end of the box and outliers show the coverage of the rest of the data. Lines are plotted at a distance of 1.5 times the interquartile range. Outliers are data with values beyond the limits of the data, if the limit of the data falls within 1.5 times the interquartile range in each direction or the limit of the box and outliers show the coverage of the rest of the data. Lines are marked by a “+” sign for each point.

10. References:


surrogates: how cross-validation errors can help us to

[31] Mack Y, Goel T, Shyy W, Haftka RT, and Queipo NV,


[34] Forrester AJ, Sobester A, and Keane AJ, “Multi-

[35] Goel T, Haftka RT, Shyy W, and Queipo NV,


[38] Lophaven SN, Nielsen HB, and Søndergaard J, *DACE – A MATLAB Kriging Toolbox*, Technical Report IMM-
TR-2002-12, Informatics and Mathematical Modelling, Technical University of Denmark, 2002.


