An Empirical Comparison of Metamodeling Strategies in Noisy Environments

Sunith Bandaru
School of Engineering Science
Skövde, Sweden
sunith.bandaru@his.se

Amos H.C. Ng
School of Engineering Science
Skövde, Sweden
amos.ng@his.se

ABSTRACT

Metamodeling plays an important role in simulation-based optimization by providing computationally inexpensive approximations for the objective and constraint functions. Additionally, metamodeling can also serve to filter noise, which is inherent in many simulation problems causing optimization algorithms to be mislead. In this paper, we conduct a thorough statistical comparison of four popular metamodeling methods with respect to their approximation accuracy at various levels of noise. We use six scalable benchmark problems from the optimization literature as our test suite. The problems have been chosen to represent different types of fitness landscapes, namely, bowl-shaped, valley-shaped, steep ridges and multi-modal, all of which can significantly influence the impact of noise. Each metamodeling technique is used in combination with four different noise handling techniques that are commonly employed by practitioners in the field of simulation-based optimization. The goal is to identify the *metamodeling strategy*, i.e. a combination of metamodeling and noise handling, that performs significantly better than others on the fitness landscapes under consideration. We also demonstrate how these results carry over to a simulation-based optimization problem concerning a scalable discrete event model of a simple but realistic production line.

CCS CONCEPTS

• **Theory of computation → Evolutionary algorithms**; • **Computing methodologies → Discrete-event simulation**; • **Mathematics of computing →** *Stochastic control and optimization*;

KEYWORDS

simulation, optimization, metamodeling, noise

ACM Reference Format:
Sunith Bandaru and Amos H.C. Ng. 2018. An Empirical Comparison of Metamodeling Strategies in Noisy Environments. In *GECCO '18: Genetic and Evolutionary Computation Conference, July 15–19, 2018, Kyoto, Japan.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3205455.3205509

1 INTRODUCTION

Simulation has been defined in the literature [22] as "the process of designing a model of a real system and conducting experiments with this model for the purpose either of understanding the behavior of the system or of evaluating various strategies (within the limits imposed by a criterion or set of criteria) for the operation of the system". The said model is often a *computational model* (e.g. finite element model, computational fluid dynamics model, discrete event simulation model, etc.) that is expensive to evaluate. The computational time for a single scenario can vary from a few seconds to a few days. When a large number of scenarios have to be evaluated, the total computational time can become infeasible. In such situations, an inexpensive approximation model of the simulation model is sought. This *model of a model* is known as a metamodel.

Simulation-Based Optimization (SBO) refers to the use of numerical optimization techniques for finding the best configuration(s) of the simulation model for minimizing (or maximizing) one or more outputs. In optimization terminology, each possible configuration is referred to as a *solution*, $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$, while the best configuration is called the *optimal solution*, $\mathbf{x}^*$. The outputs to be optimized are called the *objective/fitness functions*, $f_i(\mathbf{x})$. Often, some of the outputs may also be restricted to take certain range of values, in which case they form the *constraint functions*, $g_j(\mathbf{x})$ (inequality) and $h_k(\mathbf{x})$ (equality).

The absence of an analytical form for the objectives (or constraints), combined with the fact that real-world problems are often nonlinear, means that traditional optimization methodologies, most of which rely on function gradients, will have a hard time solving SBO problems. Evolutionary algorithms are popular in this regard due to their global search capability and ability to work without gradient information. Moreover, since evolutionary algorithms are population-based, multi-objective SBO problems can be solved to generate a set of *non-dominated solutions* [4] in a single algorithmic run. On the downside, evolutionary algorithms are known to require relatively large number of evaluations when compared to classical optimization algorithms. This makes metamodeling especially suitable to be used with evolutionary algorithms.

1.1 Noise in Simulation

Simulation models can be deterministic or stochastic. A deterministic simulation model yields the same outputs for replicated runs of a given configuration. Deterministic simulations are rare in real-world applications, and mostly appear as simplifications of stochastic simulations (also called random simulations). On the other hand, stochastic simulation models are closer to the real-world where uncertainties are prevalent. There are three main sources of noise that lead to stochasticity in simulations. Firstly, the underlying

GECCO '18, July 15–19, 2018, Kyoto, Japan

ACM ISBN 978-1-4503-5618-3/18/07…$15.00
https://doi.org/10.1145/3205455.3205509

problem may itself be stochastic. Ample examples can be found in manufacturing, transportation systems, health care, supply chain management, etc. [3]. In this case, the stochasticity is built into the simulation model using known distributions. For instance, it is common to model processing time variability in manufacturing processes using a log-normal distribution [19]. Secondly, noise can come from complex modeling techniques such as meshing in finite element analysis [10] or computational fluid dynamics [21]. Modeling noise is a function of the simulation configuration, meaning that different configurations may lead to different levels of noise. The third source of noise is the most common and is called numerical noise, arising due to pseudo-random number generation or rounding [16]. Numerical noise can be classified as white noise (independent and identically distributed), since it is independent of the configuration of the simulation.

In stochastic SBO, noise makes it impossible to obtain the true values of the objective functions, and therefore only an estimation can be made based on multiple simulation runs of the given configuration. The most common stochastic formulation uses the expected value of a *sample response function* as the estimate. Assuming that the constraints are noiseless, the SBO problem is given by [1],

$$\begin{array}{ll} \text{Minimize} & f(\mathbf{x}) = \mathbb{E}\left[F(\mathbf{x}, \xi(\omega))\right] \\ \text{Subject to} & g_j(\mathbf{x}) \geq 0 \ \ \forall\, j = 1, 2, \ldots, J \\ & h_k(\mathbf{x}) = 0 \ \ \forall\, k = 1, 2, \ldots, K \\ & \mathbf{x}^{(L)} \leq \mathbf{x} \leq \mathbf{x}^{(U)} \end{array} \tag{1}$$

The sample response function $F : \mathbb{R}^n \times \mathbb{R}^d \to \mathbb{R}$ provides a realization of $f(\mathbf{x})$ for a random vector $\xi(\omega)$. In practice, the objective function is approximated by averaging its value over a finite number of replications [5, 13], $f(\mathbf{x}) \approx \frac{1}{N}\sum_{i=1}^{N} F(\mathbf{x}, \xi_i)$, where $F(\mathbf{x}, \xi_i)$ is evaluated from a single simulation run for a random realization ξ_i of $\xi(\omega)$. Estimates other than the expected value, such as minimum, maximum and quantiles, can also be used [5].

1.2 Metamodeling in Stochastic Simulation

It is common practice in SBO studies to substitute expensive non-analytical objective functions with approximate metamodels. However, metamodeling can serve the dual purpose of smoothening out noisy fitness landscapes as pointed out in [17, 25]. Metamodels are essentially regression/interpolation techniques and therefore are capable of achieving this without severely compromising the quality of the fit. For the same reason, metamodels can sometimes also smoothen out local optima of the original fitness function without changing the location of the global optimum [13]. Severe noise can, however, cause metamodels to develop a false local optimum [12] even in relatively smooth regions of the fitness function. Hence, it is necessary to complement metamodeling methods with specific noise handling techniques. Some previous studies concerning metamodeling in noisy environments can be found in [9].

In this paper, we explore whether certain metamodeling methods have an edge over others in approximating different types of landscapes under the influence of noise over a range of problem dimensions. We also study if noise handling techniques that are commonly used in practice can improve the accuracy of these metamodels. These questions are answered both with respect to standalone metamodels and with respect to their ability in accurately representing of true optimum. The paper is organized as follows. Section 2 briefly describes the four popular metamodeling methods chosen for this paper and four simple noise handling techniques that are commonly used in SBO. Since each metamodel can be used in combination with any of the noise handling techniques, the number of *metamodeling strategies* to be compared is 16. The experimental setup and the test problem suite for carrying out the empirical comparison is described in Section 3, followed by a statistical analysis and discussion of the results in Section 4. We also illustrate how these results carry over to a scalable discrete event simulation model of a simple but realistic production line.

2 METAMODELING STRATEGIES

In this paper, we use the term 'metamodeling strategy' to refer to each unique combination of a metamodeling method with a noise handling technique. The four metamodeling methods used in this study are described next, followed by the four noise handling techniques. We assume that a set of n_s unique sample points with N replications at each sample is available for training. Each of these $n_s \times N$ samples consists of the n dependent variables $\mathbf{x}$ and a corresponding fitness realization $F(\mathbf{x}, \xi_i)$.

2.1 Metamodeling Methods

The metamodeling strategies studied in this paper use different numbers of training samples and different estimates of the fitness function for training. Therefore, for simplicity of notation, we use $T(\mathbf{x})$ to represent the target function value used for training and n_a to represent the actual number of training samples.

2.1.1 Multivariate Adaptive Regression Splines (MARS).

MARS [7] adaptively builds a metamodel of the form,

$$\hat{f}(\mathbf{x}) = a_0 + \sum_{m=1}^{M} a_m B_m^{(q)}(\mathbf{x}), \tag{2}$$

where $B_m^{(q)}(\mathbf{x})$ are multivariate spline basis functions given by,

$$B_m^{(q)}(\mathbf{x}) = \Pi_{k=1}^{K_m}[s_{km} \cdot (x_{v(k,m)} - t_{km})]_+^q, \quad s_{km} = \pm 1. \tag{3}$$

Here, K_m is the number of splits that form B_m. $v(k,m)$ represents components of $\mathbf{x}$ and t_{km} is the *knot point* on the corresponding variable. The subscript '+' indicates that the function in square brackets is a truncated power function, defined as $x_+^q = x^q$ if $x > 0$, else $x_+^q = 0$. The MARS algorithm is executed in two phases:

(1) *Forward stepwise phase:* In the forward phase, each step adds a new knot and a pair of truncated power functions to the model. The position of the knot (t_{km}) and the dimension in which to add it ($v(k,m)$) are governed by a lack-of-fit criterion, which is often the squared error loss $(\hat{f}(\mathbf{x}) - T(\mathbf{x}))^2$ obtained by performing a linear regression. The process is repeated until a user-defined maximum number of basis functions is reached.

(2) *Backward stepwise phase:* The basis functions are now deleted one at a time to prevent the model from over-fitting the training data. The first term to be deleted is the one that least affects the fit in terms of the generalized cross-validation (GCV) criterion, which trades off goodness-of-fit against

model complexity. A sequence of models, each having one less basis function than the previous one is constructed. The best model in this sequence becomes the final model.

We use a MATLAB implementation of MARS known as ARESLab [11], short for Adaptive Regression Splines toolbox for MATLAB/Octave. All parameters are set at their recommended values provided in the instruction manual.

2.1.2 *Kriging or Gaussian Process Regression (DACE).*

Kriging is an interpolation method that combines a global deterministic model with a local stochastic model. The former approximates the target function $T(\mathbf{x})$, similar to regression, while the latter allows the kriging model to interpolate the training samples by introducing local deviations. Mathematically, the kriging predictor is given by

$$\hat{f}(\mathbf{x}) = \mathcal{F}(\mathbf{a}, \mathbf{x}) + Z(\mathbf{x}), \tag{4}$$

where $\mathcal{F}(\mathbf{a}, \mathbf{x})$ can either be a constant (ordinary kriging), or more generally be a linear combination of functions (universal kriging). $Z(\mathbf{x})$ represents a Gaussian random process with zero mean, variance σ^2 and covariance,

$$Cov[Z(\mathbf{x}^{(i)}), Z(\mathbf{x}^{(j)})] = \sigma^2 \mathbf{R}[R(\boldsymbol{\theta}, \mathbf{x}^{(i)}, \mathbf{x}^{(j)})], \tag{5}$$

where, $\mathbf{R}$ is a $n_a \times n_a$ correlation matrix whose elements are given by, $R(\boldsymbol{\theta}, \mathbf{x}^{(i)}, \mathbf{x}^{(j)})$, $\boldsymbol{\theta}$ being the parameters of the correlation model.

The kriging methodology has been implemented in MATLAB as the DACE (Design and Analysis of Computer Experiments) toolbox [20]. We use the toolbox with all its default settings except for a minor enhancement described later. The parameters to be set by the user are:

(1) *regr*: Three models for $\mathcal{F}(\mathbf{a}, \mathbf{x})$ are available, (i) `regpoly0` representing a constant (ordinary kriging), (ii) `regpoly1` representing a linear regressor with $p = n + 1$ coefficients, and (iii) `regpoly2` representing a quadratic regressor with $p = (n + 2)(n + 1)/2$ coefficients.
(2) *corr*: Correlation model. Six correlation models have been implemented. The popular Gaussian correlation model is used in this paper. $R_{corrgauss}(\boldsymbol{\theta}, \mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \Pi_{k=1}^{n} e^{-\theta_k (x_k^{(i)} - x_k^{(j)})^2}$.
(3) $\boldsymbol{\theta}_0$: Initial value for $\boldsymbol{\theta}$. We use $\boldsymbol{\theta}_0 = [2\ 2\ \ldots\ 2]_n$.
(4) $\boldsymbol{\theta}_L, \boldsymbol{\theta}_U$: Lower and upper bounds for $\boldsymbol{\theta}$. We use $[0.1\ 0.1\ \ldots\ 0.1]_n$ and $[20\ 20\ \ldots\ 20]_n$, respectively.

2.1.3 *Elastic Nets (EN).*

An elastic net [26] is essentially *regularized regression.* The idea behind regularization is to use a regularization or shrinkage parameter $\lambda \geq 0$ to penalize overly complex models. Two regularization methods are commonly used. The L_2 regularization method penalizes the sum of squares of the coefficients and is known as ridge regression. The L_1 regularization method penalizes the sum of absolute values of the coefficients and is known as LASSO (Least Absolute Shrinkage and Selection Operator). The advantage of LASSO is that it can completely eliminate some of the variables by zeroing out the corresponding coefficients, thus giving a parsimonious model.

Elastic nets combine ridge and LASSO regression through a parameter $\alpha \in [0, 1]$. The cost function for elastic net regression is,

$$\sum_{i=1}^{n_a} (T(\mathbf{x}^{(i)}) - \hat{f}(\mathbf{x}^{(i)}))^2 + \lambda \left[\frac{(1-\alpha)}{2} ||\mathbf{a}||_2^2 + \alpha ||\mathbf{a}||_1 \right], \tag{6}$$

where $\hat{f}$ is the standard quadratic regression model with coefficients $\mathbf{a}$, $||\mathbf{a}||_2^2 = \sum_{j=1}^{p} a_j^2$ and $||\mathbf{a}||_1 = \sum_{j=1}^{p} |a_j|$. With $\lambda = 0$, the above reduces to ordinary least squares estimation. $\alpha = 0$ corresponds to ridge regression and $\alpha = 1$ gives LASSO.

We use the GLMNET MATLAB toolbox [8] with its default settings in this paper.

2.1.4 *Random Forests (RF).*

Random forests are a variation of the bagging approach proposed in [2]. Bagging or bootstrap aggregation is a method for reducing the variance of predictors by training multiple low bias models on bootstrap samples (random samples with replacement). Bootstrap samples have the same size as the original training set. The prediction (for regression) is obtained by averaging the output over all the trained models. The bagged tree ensemble uses a user-defined number of *unpruned* regression trees. Random forests provide an improvement over bagged trees by additionally performing *feature bagging*. A random sample of $NVarToSample < n$ variables are considered at each split instead of considering all n variables. Typically, $NVarToSample = \lfloor n/3 \rfloor$ for regression. Like the bagged tree ensemble, random forests also use a user-defined number of *unpruned* regression trees, *ntrees*. Feature bagging and bootstrapping help in de-correlating the trees. We use MATLAB's `TreeBagger` function in this paper with $ntrees = 100$, $NVarToSample = \lfloor n/3 \rfloor$, $MinLeaf = 5$ and $prune =$`false`.

2.2 Noise Handling Techniques

There are three broad ways of dealing with noise in general [23], (i) leave noise in, (ii) filter noise out, (iii) correct for noise. The following noise handling techniques use one or more of these approaches.

2.2.1 *Mean Over Replications.*

This is the simplest and most common way of correcting for noise in simulation. The N replications of each unique sample are averaged to give an estimate of $f(\mathbf{x})$ as shown earlier. Thus, $T(\mathbf{x}) = \frac{1}{N} \sum_{i=1}^{N} F(\mathbf{x}, \xi_i)$. The number of training samples here is $n_a = n_s$.

2.2.2 *Median Over Replications.*

The median (second quantile) of the N replications can also be used as an estimate of $f(\mathbf{x})$ to correct for noise at each unique sample location. Thus, $T(\mathbf{x}) = \text{median}(F(\mathbf{x}, \xi_1), F(\mathbf{x}, \xi_2), \ldots F(\mathbf{x}, \xi_N))$. It is well known that the median is less sensitive to outliers than the mean, so it may lead to better metamodels. The number of training samples is again $n_a = n_s$.

2.2.3 *All Replications.*

All N replications of each unique sample can be used for training, leaving the noise untouched. The actual number of training samples with this technique is $n_a = n_s \times N$, and the target is $T(\mathbf{x}) = F(\mathbf{x}, \xi_i) \forall\ i$.

2.2.4 *Mean Over Filtered Replications.*

Outliers among the N replications can be identified using a rule-of-thumb such as $\mu \pm k\ \sigma$. The average calculated after removing the outliers may provide a more stable estimate of $f(\mathbf{x})$ for each unique sample. Therefore, $T(\mathbf{x}) = \frac{1}{|C|} \sum_{i \in C} F(\mathbf{x}, \xi_i)$, where

$$C = \{i : \mu - k\ \sigma \leq F(\mathbf{x}, \xi_i) \leq \mu + k\ \sigma\},$$

μ and σ being the mean of standard deviation of $F(\mathbf{x}, \xi_i)$ over all i. The number of training samples is $n_a = n_s$. We use $k = 1$ in this paper.

3 EXPERIMENTAL SETUP

As mentioned in the previous sections, we empirically compare 16 metamodeling strategies, resulting from the combinations of metamodeling methods with noise handling techniques. These strategies are:

(1) MARS with mean over replications, $MARS_\mu$
(2) MARS with median over replications, $MARS_M$
(3) MARS with all replications, $MARS_A$
(4) MARS with mean over filtered replications, $MARS_F$
(5) DACE with mean over replications, $DACE_\mu$
(6) DACE with median over replications, $DACE_M$
(7) DACE with all replications, $DACE_A$
(8) DACE with mean over filtered replications, $DACE_F$
(9) EN with mean over replications, EN_μ
(10) EN with median over replications, EN_M
(11) EN with all replications, EN_A
(12) EN with mean over filtered replications, EN_F
(13) RF with mean over replications, RF_μ
(14) RF with median over replications, RF_M
(15) RF with all replications, RF_A
(16) RF with mean over filtered replications, RF_F

Each metamodeling strategy is tested on,

(1) six benchmark problems with different types of fitness landscapes as shown in Table 1. Though none of the problems are simulation models, we emulate simulation by treating the test functions as *black boxes*, i.e., the metamodeling strategies do not use the function definitions in any way.
(2) five different problem dimensions, $n = \{10, 20, 30, 40, 50\}$
(3) five different noise levels, $\beta = \{0, 0.001, 0.01, 0.1, 1\}$
We use Gaussian multiplicative noise as specified in [6] and given by,

$$F(\mathbf{x}, \xi(\omega); \beta) = f(\mathbf{x}) \exp(\beta \mathcal{N}(0, 1)). \tag{7}$$

β controls the strength of noise, with $\beta = 0$ being noiseless, $\beta = 0.01$ corresponding to moderate noise and $\beta = 1$ corresponding to severe noise. The Gaussian noise model is scale invariant and results in a log-normally distributed noise.

Thus, there are a total of $6 \times 5 \times 5 = 150$ experiments for each strategy. Each of these experiments is run 10 times with different random seeds to account for statistical variation and the median performance metric is presented.

For every run of each experiment, both training and test samples are independently generated using Latin hypercube sampling in the domain $[-5, 5]^n$. The number of unique training samples is scaled with the number of variables as $n_s = 15n^1$. Each training sample is evaluated (replicated) $N = 10$ times to account for different levels of noise, thus giving a total of $n_s \times N$ samples. As discussed in Section 2.2, each metamodeling strategy uses these replications differently. For a given experiment and a given run, all metamodeling strategies use the same unique samples and their replications. The same is true for the test samples.

3.1 Performance Metrics

The number of test samples is fixed at $n_t = 100$ for all experiments. It is important to note that the test samples are noiseless, so that the metamodel accuracy is measured with respect to true function values. The following performance metrics are used:

(1) Normalized Root Mean Squared Error (*NRMS*): This is a standard error measure. Normalization ensures that the values can be compared across various noise levels and problem dimensions for a given problem. It is given by,

$$NRMS = \frac{1}{\sqrt{n_t}} \frac{\sqrt{\sum_{i=1}^{n_t} (f(\mathbf{x}^{(i)}) - \hat{f}(\mathbf{x}^{(i)}))^2}}{(f_{max} - f_{min})},$$

where,

$$f_{max} = \max(f(\mathbf{x}^{(1)}), f(\mathbf{x}^{(2)}), \ldots, f(\mathbf{x}^{(n_t)}))$$
$$f_{min} = \min(f(\mathbf{x}^{(1)}), f(\mathbf{x}^{(2)}), \ldots, f(\mathbf{x}^{(n_t)}))$$

(2) Rank Error (*RE*): Evolutionary algorithms are comparison-based optimizers which means that they select good solutions through pairwise comparisons. Therefore, when metamodels are to be used with evolutionary algorithms, they only have to be accurate enough to select the better of two solutions. In other words, they need not have low *NRMS* values to be useful. We therefore use a metric called the *rank error*, defined in [14] and given by,

$$RE = \frac{SwappedPairs}{\binom{n_t}{2}},$$

where, *SwappedPairs* represents the number of pairs of test samples for which the order with respect to $\hat{f}(\mathbf{x})$ (predicted by the metamodel) does not agree with the true order (i.e., with respect to $f(\mathbf{x})$).
(3) Optimum Location Error ($\Delta\mathbf{x}$ and Δf): In addition to approximating the fitness landscape well, a metamodel should be able to represent the true optimum accurately, both in variable and objective space [15]. Let $\hat{\mathbf{x}}^*$ be the optimum found by an evolutionary algorithm when using metamodel-based optimization. The optimum location errors are defined as,

$$\Delta\mathbf{x} = \|\hat{\mathbf{x}}^* - \mathbf{x}^*\| \text{ and } \Delta f = |f(\hat{\mathbf{x}}^*) - f(\mathbf{x}^*)|.$$

4 RESULTS AND DISCUSSION

4.1 Statistical Analysis of *NRMS* and *RE*

The first part of the results concerns performance metrics *NRMS* and *RE* for standalone metamodeling strategies. We first illustrate the statistical analysis of *NRMS* on the Rosenbrock function (Problem 3 in Table 1) for $n = 20$ variables. The median *NRMS* values obtained from 10 repeated runs of this problem instance for different metamodeling strategies and at different levels of noise are shown in Table 2. For each of the five experiments (columns), a Kruskal-Wallis rank-sum test [18] is performed at $\alpha = 0.05$ level of significance to test the null hypothesis that there is no significant difference between the metamodeling strategies. We get p-values of 1.12×10^{-13}, 1.19×10^{-13}, 1.55×10^{-14}, 5.92×10^{-15} and 1.47×10^{-22} for the five columns from left to right. Thus, the null hypothesis can be rejected in all cases and it can be concluded that at least two of the metamodeling strategies are significantly different from

[1] Note that the number coefficients to be estimated in a quadratic regression model is $p = (n+2)(n+1)/2$. Thus, for problem dimensions $n = \{10, 20\}$, the quadratic regression problem is *overdetermined* ($p < n_s$), whereas it is *underdetermined* ($p > n_s$) for $n = \{30, 40, 50\}$. This helps us test both scenarios.

Table 1: Optimization benchmarks used in this study.

ID	Name	Function Definition	Landscape	Optimum
1	Sphere	$f(\mathbf{x}) = \sum_{i=1}^{n} x_i^2$	Bowl	$\mathbf{x}^* = \mathbf{0}$, $f(\mathbf{x}^*) = 0$
2	Rotated ellipsoid	$f(\mathbf{x}) = \sum_{i=1}^{n} \sum_{j=1}^{i} x_j^2$	Bowl	$\mathbf{x}^* = \mathbf{0}$, $f(\mathbf{x}^*) = 0$
3	Rosenbrock	$f(\mathbf{x}) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	Valley	$\mathbf{x}^* = \mathbf{1}$, $f(\mathbf{x}^*) = 0$
4	Michalewicz	$f(\mathbf{x}) = -\sum_{i=1}^{n} \sin(x_i) \sin^{2m}\left(ix_i^2/\pi\right)$	Steep ridges	$\mathbf{x}^*$ = See [24], $f(\mathbf{x}^*) = -0.99864n + 0.30271$ [24]
5	Rastringin	$f(\mathbf{x}) = 10n + \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i)]$	Multi-modal	$\mathbf{x}^* = \mathbf{0}$, $f(\mathbf{x}^*) = 0$
6	Ackley	$f(\mathbf{x}) = -20\exp(-0.2\sqrt{\sum_{i=1}^{n} x_i^2/n})$ $- \exp(\sum_{i=1}^{n} \cos(2\pi x_i)/n) + 20 + \exp(1)$	Multi-modal	$\mathbf{x}^* = \mathbf{0}$, $f(\mathbf{x}^*) = 0$

Table 2: Median *NRMS* values from 10 runs of all metamodeling strategies at different noise levels for 20-dimensional Rosenbrock function.

Strategy		Noise levels				
ID	Name	$\beta = 0$	$\beta = 0.001$	$\beta = 0.01$	$\beta = 0.1$	$\beta = 1$
1	$MARS_\mu$	0.1660	0.1793	0.1801	0.1803	0.4810
2	$MARS_M$	0.1660	0.1793	0.1801	0.1817	**0.2295**
3	$MARS_A$	0.1660	0.1813	0.1783	0.1740	0.4729
4	$MARS_F$	0.1660	0.1793	0.1750	0.1738	**0.2511**
5	$DACE_\mu$	**0.1500**	**0.1434**	**0.1373**	**0.1472**	0.9053
6	$DACE_M$	**0.1500**	**0.1434**	**0.1374**	**0.1477**	0.5243
7	$DACE_A$	**0.1500**	**0.1434**	**0.1373**	**0.1472**	0.9053
8	$DACE_F$	**0.1500**	**0.1434**	**0.1377**	**0.1504**	0.5091
9	EN_μ	**0.0837**	**0.0897**	**0.0846**	**0.0847**	0.4288
10	EN_M	**0.0835**	**0.0889**	**0.0853**	**0.0867**	**0.1776**
11	EN_A	**0.0829**	**0.1352**	**0.1316**	**0.1380**	0.4325
12	EN_F	**0.0835**	**0.0881**	**0.0834**	**0.0856**	**0.1852**
13	RF_μ	0.1537	0.1672	0.1655	0.1754	**0.4134**
14	RF_M	0.1546	0.1675	0.1665	0.1742	**0.1595**
15	RF_A	0.1539	0.1659	0.1631	0.1675	0.4329
16	RF_F	0.1533	0.1682	0.1656	0.1752	**0.1725**

each other in each column. In order to identify the best strategies for each experiment, post-hoc multiple comparisons tests must be performed. We use MATLAB’s `multcompare` function to perform the Tukey-Kramer test, which reveals all groups of statistically indistinguishable strategies. The strategies that are found to be equivalent to the best median strategy are shown in bold for each column in Table 2.

In order to summarize the results for Rosenbrock function with $n = 20$ variables, we can define a *significance score* over the five experiments. The number of times a particular metamodeling strategy’s median *NRMS* appears in bold in Table 2 is a measure of how successful the strategy is overall. We observe that strategies 10 and 12 (EN_M and EN_F) are among the best strategies 5 out of 5 times.

We can now extend this analysis over all problems and dimensions. To better visualize the *NRMS* values in a concise form and possibly detect patterns, we propose to use what we call *Statistical Error Maps* or SEMs. The SEM of median *NRMS* at $\beta = 0$ for all problems and dimensions is shown in Figure 1. The colorbar on the right indicates the range of *NRMS*. The rows of the map are metamodeling strategies denoted by their IDs, and the columns are experiments grouped by the problems. Within each ’Problem’ block, the columns from left to right correspond to problem dimensions $n = 10, 20, 30, 40$ and 50, respectively. The black markers indicate statistical indistinguishability of the strategies to the best median strategy from corresponding columns. Notice that no markers appear in Problem 4 (Michalewicz) for $n = 30, 40$ and 50. This happens when the Kruskal-Wallis null hypothesis cannot be rejected, and therefore no post-hoc analysis can be performed.

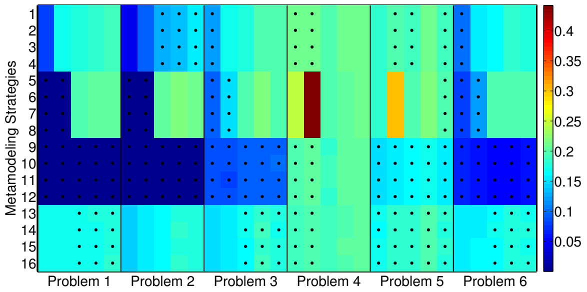
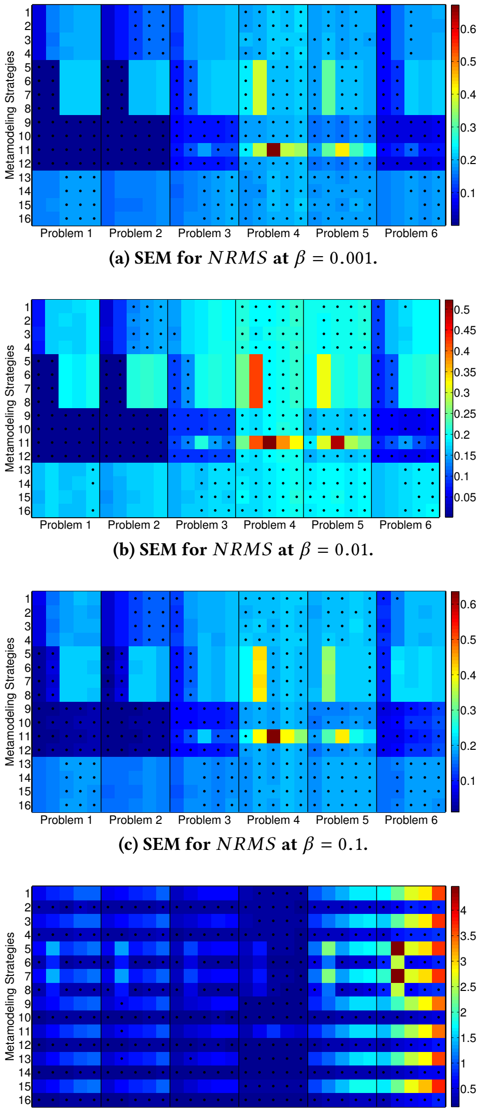


Figure 1: Statistical Error Map for *NRMS* at $\beta = 0$.

In about the same amount of space, the SEM conveys much more information than Table 2. Moreover, interesting patterns can be detected by visual inspection. For example, notice that the performance of DACE based strategies (with IDs 5, 6, 7 and 8) drop significantly between $n = 20$ and $n = 30$ for Problems 1, 2, 3 and 6. The reason for this is that our DACE variant switches from `regpoly2` to `regpoly1` to avoid an underdetermined regression problem for $n = 30, 40$ and 50. We also note that all strategies perform poorly on Problem 4 (Michalewicz) which features steep ridges. There is no trend indicating that the metamodel accuracy deteriorates with problem dimension. This indicates that the scaling factor used for unique training samples, i.e. $n_s = 15n$, is adequate.

Similar SEMs generated for $\beta = 0.001, 0.01, 0.1$ and 1 are shown in Figures 2a-2d. The patterns discussed above can still be seen in these figures, though to a much less extent for $\beta = 1$. The corresponding noise level has been classified as ‘severe’ in [6]. It is apparent from the range of *NRMS* values in Fig. 2d that this level of noise adversely affects the metamodels. Note that for $\beta = 0, 0.001, 0.01$ and 0.1, no significant differences can be observed among the noise



(a) SEM for $NRMS$ at $\beta = 0.001$.

(b) SEM for $NRMS$ at $\beta = 0.01$.

(c) SEM for $NRMS$ at $\beta = 0.1$.

(d) SEM for $NRMS$ at $\beta = 1$.

Figure 2: Statistical Error Map (SEM) for $NRMS$.

handling techniques for any of the metamodeling methods. However, they matter when the noise is severe, as seen in Fig. 2d. Noise handling by 'mean over replications' and 'all replications' are rarely among the best performing strategies.

From a cursory look, it can already be seen that elastic net based strategies outperform other methods. However, to quantitatively rank all the strategies, we can calculate significance scores as before. Table 3 shows the break-up of these scores over different noise levels. All scores are out of 30, the number of experiments at each noise level. The overall score is out of 150. The scores indicate that EN_M and EN_F are the best overall metamodeling strategies with respect to $NRMS$. For low to moderate amount of noise, all EN based strategies are found to be better than others. EN_μ comes in second due to its poor performance under severe noise. Surprisingly, RF_M, RF_F, $MARS_M$ and $MARS_F$ beat all DACE based strategies. We attribute it to the use of linear regressor in DACE.

Table 3: Significance scores for $NRMS$ broken-up by noise level.

Strategy		Noise levels					Overall
ID	Name	$\beta = 0$	$\beta = 0.001$	$\beta = 0.01$	$\beta = 0.1$	$\beta = 1$	Score
1	$MARS_\mu$	10	13	14	14	4	55
2	$MARS_M$	10	13	13	13	30	79
3	$MARS_A$	10	15	14	13	4	56
4	$MARS_F$	10	13	13	12	30	78
5	$DACE_\mu$	9	15	11	11	3	49
6	$DACE_M$	9	15	11	11	23	69
7	$DACE_A$	9	15	11	11	3	49
8	$DACE_F$	9	15	11	11	23	69
9	EN_μ	27	30	30	30	6	123
10	EN_M	27	30	30	30	30	147
11	EN_A	27	21	20	21	1	90
12	EN_F	27	30	30	30	30	147
13	RF_μ	16	19	17	20	7	79
14	RF_M	16	19	17	20	30	102
15	RF_A	16	18	15	20	5	74
16	RF_F	16	19	17	20	30	102

A similar analysis can be done for rank error RE. Skipping over individual SEMs, we directly show the significance scores (out of 30) for RE at different noise levels in Table 4. It is immediately clear that RE and $NRMS$ are not equivalent or correlated. Due to lower fidelity requirement when purely used for comparisons, it is easier for metamodels to be good at RE than at $NRMS$. For example, RF based strategies are now at par with EN based strategies. Also, noise handling techniques seem to be irrelevant when considering RE. EN_A is found to be the best metamodeling strategy. Contrary to what was observed with $NRMS$, DACE based strategies have a slight edge here over MARS based ones. On the other hand, severe noise has the same effect on RE as it had on $NRMS$.

4.2 Statistical Analysis of Δx and Δf

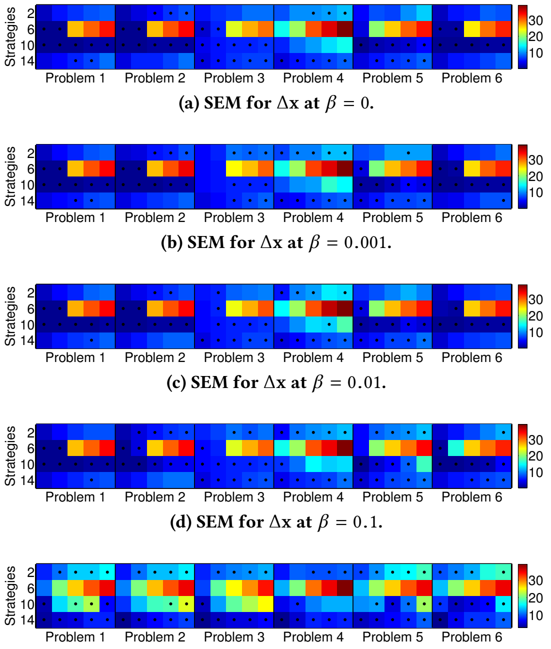
Having identified the best standalone metamodeling strategies, we now use them to optimize the functions. Metamodels can be used in two broad ways to assist optimization, direct fitness replacement and indirect fitness replacement. Direct fitness replacement methods use the approximations from the metamodel in place of actual function evaluation. This can be done in three ways: (a) without evolution control, (b) fixed evolution control, and (iii) adaptive evolution control. Evolution control basically specifies when and how the metamodel approximations should be used and the metamodels should be updated. When no evolution control is used, the metamodel is static and all fitness values are exclusively obtained from the metamodel. Fixed evolution control uses fixed heuristics/strategies to answer the when and the how. Adaptive evolution control is data-driven. It monitors the progress of the optimization and automatically switches back and forth between the metamodel and the true function.

Table 4: Significance scores for *RE* broken-up by noise level.

Strategy		Noise levels					Overall
ID	Name	$\beta = 0$	$\beta = 0.001$	$\beta = 0.01$	$\beta = 0.1$	$\beta = 1$	Score
1	$MARS_\mu$	9	11	10	11	13	54
2	$MARS_M$	9	10	12	11	16	58
3	$MARS_A$	9	11	11	11	14	56
4	$MARS_F$	9	11	12	8	20	60
5	$DACE_\mu$	13	14	14	13	11	65
6	$DACE_M$	13	14	14	12	10	63
7	$DACE_A$	13	14	14	13	11	65
8	$DACE_F$	13	14	14	12	11	64
9	EN_μ	25	26	27	25	8	111
10	EN_M	25	25	27	25	13	115
11	EN_A	25	28	27	26	16	122
12	EN_F	25	25	27	25	14	116
13	RF_μ	22	21	19	21	30	113
14	RF_M	22	21	20	20	30	113
15	RF_A	23	16	16	19	30	104
16	RF_F	22	20	19	21	30	112

In this section, we use the metamodel without any evolution control, which is possibly the worst way to use metamodels in optimization. Our purpose here is only to study the performance of the metamodeling strategies in accurately representing the true optimum. The use of an evolution control, whether fixed or adaptive, might affect the metamodeling strategies differently. Therefore, it is best to keep them out of the picture. Moreover, there is no consensus on the what constitutes an optimal evolution control.

Based on the results from the previous section, we choose strategies that use 'median over replications' for noise handling. These are $MARS_M$, $DACE_M$, EN_M and RF_M (with IDs 2, 6, 10 and 14). The experimental procedure is exactly the same as before, except that after training, we optimize the static metamodel using MATLAB's ga function. The parameters for this GA are: (i) population size = $5n$, (ii) max generations = $20n$. Other parameters take their default values. The best solution $\hat{\mathbf{x}}^*$ found by the GA is used to calculate the optimum location errors $\Delta\mathbf{x}$ and Δf. The SEMs for $\Delta\mathbf{x}$ at different noise levels are shown in Figures 3. Δf requires true function values for all $\hat{\mathbf{x}}^*$. Once evaluated, the overall significance scores for both $\Delta\mathbf{x}$ and Δf are calculated as explained before for *NRMS* and *RE*. These scores (out of 150) are shown in Table 5. Additionally, the *combined score* (also out of 150) is also shown. An experiment counts towards the combined score of a strategy, if and only if both $\Delta\mathbf{x}$ and Δf are statistically indistinguishable from their corresponding best median strategies. In other words, the black marker must appear at corresponding places in the SEMs of both $\Delta\mathbf{x}$ and Δf.



(a) SEM for $\Delta\mathbf{x}$ at $\beta = 0$.

(b) SEM for $\Delta\mathbf{x}$ at $\beta = 0.001$.

(c) SEM for $\Delta\mathbf{x}$ at $\beta = 0.01$.

(d) SEM for $\Delta\mathbf{x}$ at $\beta = 0.1$.

(e) SEM for $\Delta\mathbf{x}$ at $\beta = 1$.

Figure 3: Statistical Error Map (SEM) for $\Delta\mathbf{x}$.

Table 5: Significance scores for $\Delta\mathbf{x}$ and Δf.

Strategy		Overall Score		Combined Score
ID	Name	$\Delta\mathbf{x}$	Δf	(Not summation)
2	$MARS_M$	65	53	48
6	$DACE_M$	28	42	25
10	EN_M	115	118	113
14	RF_M	97	92	90

4.3 Discrete Event Simulation Model

Next, we consider a discrete event simulation model of a simple production line (called an unpaced flow line) as shown in Figure 4 consisting of $S = 5$ workstations with $S - 1 = 4$ inter-station buffers. This model has the advantage of being scalable in S as well as being realistic enough to be applicable in many industrial settings. The throughput of the line is governed by workstation availability (α_i %), processing time (β_i seconds) and repair time (γ_i seconds) of the workstations/machines, which can take any of the following discrete values:

$$\begin{aligned} \alpha_i &= \{90, 92, 94, 96, 98\} && \forall\ i = 1, 2, \ldots S \\ \beta_i &= \{60, 65, 70, 75, 80\} && \forall\ i = 1, 2, \ldots S \\ \gamma_i &= \{180, 210, 240, 270, 300\} && \forall\ i = 1, 2, \ldots S \end{aligned} \tag{8}$$

The capacities of inter-station buffers can take integer values $B_i = \{1, 2, \ldots 10\}$. The processing times are assumed to be constant, which is realistic for automated machining processes. The time-to-failure of the workstations are modeled with exponential distributions and the randomness of repair times is modeled using Erlang distributions. Thus, the throughput of the production line is a stochastic function of $4S - 1$ variables α_i, β_i, γ_i and B_i. The corresponding SBO problem requires the throughput to be maximized, which by itself can easily be achieved by setting α's and B's as high as possible and β's and γ's as low as possible. However, assuming that currently each machine has $\alpha^{orig} = 90$, $\beta^{orig} = 80$ and $\gamma^{orig} = 300$, every step of improvement in any of these variables

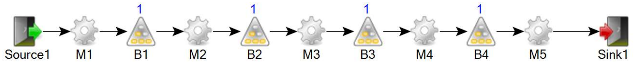


Figure 4: Discrete event simulation model representing an unpaced production flow line.

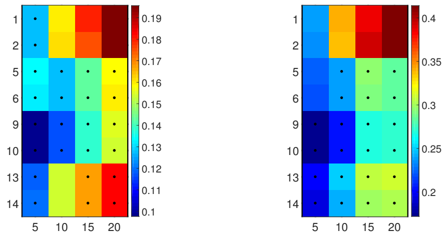


Figure 5: SEM for *NRMS* (left) and *RE* (right) of discrete event simulation model shown in Figure 4.

requires an investment. Thus, minimization of the total investment gives a trade-off with respect to throughput maximization.

As in the case of the test problems, we will assess the performance of the metamodeling strategies in approximating the throughput for increasing number of machines, i.e. $S = \{5, 10, 15, 20\}$. The corresponding increase in the number of variables is $n = \{19, 39, 59, 79\}$. The experimental setup remains the same as in Section 3, except that here we only consider the mean and median based strategies, i.e. those with IDs 1, 2, 5, 6, 9, 10, 13 and 14. Note that the noise level β need not be specified as the problem is inherently stochastic. However, we have estimated the β to be approximately 0.01, which corresponds to a moderate noise level according to the Guassian multiplicative noise model.

Figure 5 shows the SEMs for *NRMS* and *RE*. Note that DACE can use the quadratic regressor for only $S = 5$ ($n = 19$). For other problem dimensions, it uses a linear regressor like before. In terms of the range and variation of error values, the SEMs look similar to those of the rotated ellipsoid function, which suggests that the fitness landscape of throughput function may resemble a bowl. The performance score of EN is superior (4 out of 4 in both *NRMS* and *RE*) to the scores of other methods. However, there is no significant difference between the noise handling techniques. This conclusion was also drawn earlier for low and moderate noise levels on the test problems.

5 CONCLUSIONS

We performed an empirical comparison of 16 metamodeling strategies in noisy environments with different levels of noise using scalable test functions with characteristic fitness landscapes. The metamodeling strategies are combinations of four metamodeling methods with four common noise handling techniques. We developed a statistical test procedure for identifying the best strategies for different experiments and a visualization technique called 'statistical error maps' for concisely presenting the performance metrics. Four performance metrics were used to compare the metamodeling strategies. We defined 'significance scores' to summarize the results over multiple experiments. Among the four metamodeling methods, elastic net was found to be the best performing with respect to all metrics. Among the noise handling techniques, 'median over replications' and 'mean over filtered replications' outperformed others in severe noise conditions. However, under low and moderate noise 'median over replications' and 'mean over replications' are statistically indistinguishable. Our recommendation is therefore to use elastic nets as the metamodeling method with 'median over replications' for noise handling when solving simulation based optimization problems.

REFERENCES

[1] Satyajith Amaran, Nikolaos V Sahinidis, Bikram Sharda, and Scott J Bury. 2014. Simulation optimization: a review of algorithms and applications. *4OR* 12, 4 (2014), 301–333.

[2] Leo Breiman. 1996. Bagging predictors. *Machine learning* 24, 2 (1996), 123–140.

[3] Chun-hung Chen and Loo Hay Lee. 2011. *Stochastic simulation optimization: an optimal computing budget allocation.* Vol. 1. World scientific.

[4] K Deb. 2001. *Multi-objective optimization using evolutionary algorithms.* New York: Wiley.

[5] Geng Deng and Michael C Ferris. 2009. Variable-number sample-path optimization. *Mathematical Programming* 117, 1 (2009), 81–109.

[6] Steffen Finck, Nikolaus Hansen, Raymond Ros, and Anne Auger. 2010. *Real-parameter black-box optimization benchmarking 2010: Presentation of the noisy functions.* Technical Report. Technical Report 2009/21, Research Center PPE.

[7] J H Friedman. 1991. Multivariate adaptive regression splines. *The Annals of Statistics* 19, 1 (1991), 1–67.

[8] Jerome H. Friedman, T Hastie, and Rob Tibshirani. 2010. Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of statistical software* 33, 1 (2010), 1–22. arXiv:arXiv:1501.0228

[9] Alexander I J. Forrester, Andy J Keane, and Neil W Bressloff. 2006. Design and analysis of "Noisy" computer experiments. *AIAA journal* 44, 10 (2006), 2331–2339.

[10] Stefan Jakobsson, Michael Patriksson, Johan Rudholm, and Adam Wojciechowski. 2010. A method for simulation based optimization using radial basis functions. *Optimization and Engineering* 11, 4 (2010), 501–532.

[11] G Jekabsons. 2009. ARESLab: Adaptive regression splines toolbox for MATLAB. (2009).

[12] Yaochu Jin. 2005. A comprehensive survey of fitness approximation in evolutionary computation. *Soft computing* 9, 1 (2005), 3–12.

[13] Yaochu Jin and Jürgen Branke. 2005. Evolutionary optimization in uncertain environments-a survey. *IEEE Transactions on evolutionary computation* 9, 3 (2005), 303–317.

[14] Thorsten Joachims. 2005. A Support Vector Method for Multivariate Performance Measures. In *Proceedings of the 22nd International Conference on Machine Learning ICML'05.* 377–384. DOI:http://dx.doi.org/10.1145/1102351.1102399

[15] Anthony C Keys, Loren Paul Rees, and Allen G Greenwood. 2002. Performance measures for selection of metamodels to be used in simulation optimization. *Decision Sciences* 33, 1 (2002), 31–58.

[16] Jack PC Kleijnen. 2012. Design and analysis of Monte Carlo experiments. In *Handbook of Computational Statistics.* Springer, 529–547.

[17] PN Koch, R-J Yang, and Lei Gu. 2004. Design for six sigma through robust optimization. *Structural and Multidisciplinary Optimization* 26, 3-4 (2004), 235–248.

[18] William H Kruskal and W Allen Wallis. 1952. Use of ranks in one-criterion variance analysis. *Journal of the American statistical Association* 47, 260 (1952), 583–621.

[19] Averill M Law, W David Kelton, and W David Kelton. 1991. *Simulation modeling and analysis.* Vol. 2. McGraw-Hill New York.

[20] S. N. Lophaven, H. B. Nielsen, and Jacob Søndergaard. 2002. *Aspects of the Matlab toolbox DACE.* Technical Report. Technical Univeristy of Denmark, Lyngby, Denmark. 1–23 pages. http://www.imm.dtu.dk/~hbni/dace/

[21] Jens I Madsen, Wei Shyy, and Raphael T Haftka. 2000. Response surface techniques for diffuser shape optimization. *AIAA journal* 38, 9 (2000), 1512–1518.

[22] E Shannon Robert. 1975. *Systems simulation: The art and science.* Prentice-Hall, Englewood Cliffs, New Jersey.

[23] Choh-Man Teng. 2001. A Comparison of Noise Handling Techniques. In *FLAIRS Conference.* 269–273.

[24] Charlie Vanaret, Jean-Baptiste Gotteland, Nicolas Durand, and Jean-Marc Alliot. 2014. Certified Global Minima for a Benchmark of Difficult Optimization Problems. (2014). https://hal-enac.archives-ouvertes.fr/hal-00996713 Preprint.

[25] G Gary Wang and Songqing Shan. 2007. Review of metamodeling techniques in support of engineering design optimization. *Journal of Mechanical design* 129, 4 (2007), 370–380.

[26] H Zou and T Hastie. 2005. Regularization and variable selection via the elastic-net. *Journal of the Royal Statistical Society* 67 (2005), 301–320.