Journal of Machine Learning Research 10 (2009) 2039-2078 Submitted 12/08; Revised 3/09; Published 9/09

Evolutionary Model Type Selection for Global Surrogate Modeling

Dirk Gorissen DIRK.GORISSEN@UGENT.BE
Tom Dhaene TOM.DHAENE@UGENT.BE
Filip De Turck FILIP.DETURCK@UGENT.BE
Ghent University - IBBT
Department of Information Technology (INTEC)
Gaston Crommenlaan 8 bus 201
9050 Gent, Belgium

Editor: Melanie Mitchell

Abstract

Due to the scale and computational complexity of currently used simulation codes, global surrogate (metamodels) models have become indispensable tools for exploring and understanding the design space. Due to their compact formulation they are cheap to evaluate and thus readily facilitate visualization, design space exploration, rapid prototyping, and sensitivity analysis. They can also be used as accurate building blocks in design packages or larger simulation environments. Consequently, there is great interest in techniques that facilitate the construction of such approximation models while minimizing the computational cost and maximizing model accuracy. Many surrogate model types exist (Support Vector Machines, Kriging, Neural Networks, etc.) but no type is optimal in all circumstances. Nor is there any hard theory available that can help make this choice. In this paper we present an automatic approach to the model type selection problem. We describe an adaptive global surrogate modeling environment with adaptive sampling, driven by speciated evolution. Different model types are evolved cooperatively using a Genetic Algorithm (heterogeneous evolution) and compete to approximate the iteratively selected data. In this way the optimal model type and complexity for a given data set or simulation code can be dynamically determined. Its utility and performance is demonstrated on a number of problems where it outperforms traditional sequential execution of each model type.

Keywords: model type selection, genetic algorithms, global surrogate modeling, function approximation, active learning, adaptive sampling

1. Introduction

For many problems from science and engineering it is impractical to perform experiments on the physical world directly (e.g., airfoil design, earthquake propagation). Instead, complex, physics-based simulation codes are used to run experiments on computer hardware. While allowing scientists more flexibility to study phenomena under controlled conditions, computer experiments require a substantial investment of computation time. One simulation may take many minutes, hours, days or even weeks. A simpler approximation of the simulator is needed to make sensitivity analysis, visualization, design space exploration, etc. feasible (Forrester et al., 2008; Simpson et al., 2008).

As a result researchers have turned to various approximation methods that mimic the behavior of the simulation model as closely as possible while being computationally cheap(er) to evaluate. Different types of approximation methods exist, each with their relative strengths. This work concentrates on the use of data-driven, global approximations using compact surrogate models (also known as emulators, metamodels or response surface models) in the context of computer experiments. The objective is to construct a high fidelity approximation model that is as accurate as possible over the *complete* design space of interest using as little simulation points as possible. Once constructed, the global surrogate model (also referred to as a replacement metamodel[1]) is reused in other stages of the computational science and engineering pipeline. So optimization is not the main goal, but rather a useful post processing step.

The primary users of global surrogate modeling methods are domain experts, few of which will be experts in the intricacies of efficient sampling and modeling strategies. Their primary concern is obtaining an accurate replacement metamodel for their problem as fast as possible and with minimal overhead. Model (type) selection, model parameter optimization, sampling strategy, etc. are of lesser or no interest to them. Thus, this paper explores an automated way to help answer the always recurring question from domain experts *"Which approximation method is best for my data?"*. An evolutionary algorithm is presented that combines automatic model *type* selection, automatic model parameter optimization, and sequential design exploration.

In the next Section we describe the problem of global surrogate modeling followed by an in depth discussion of the motivation for this work in Section 3. The core approach presented in this paper is discussed in Section 5 followed by a critical analysis in Section 6. Section 7 describes a number of surrogate modeling problems we shall use to demonstrate the proposed approach, followed by their discussion in Section 10 (the experimental setup is described in Section 9). We conclude in Section 12 with pointers to future work.

2. Global Surrogate Modeling

The mathematical formulation of the problem is as follows: approximate an unknown multivariate function $f : \Omega \mapsto \mathbb{C}^n$, defined on some domain $\Omega \subset \mathbb{R}^d$, whose function values $f(X) = \{f(x_1), ..., f(x_k)\} \subset \mathbb{C}^n$ are known at a fixed set of pairwise distinct sample points $X = \{x_1, ..., x_k\} \subset \Omega$. Constructing an approximation requires finding a suitable function s from an approximation space S such that $s : \Omega \mapsto \mathbb{C}^n \in S$ and s closely resembles f as measured by some criterion ξ, where ξ constitutes three parts:

$$\xi = (\Lambda, \varepsilon, \tau).$$

Λ is the generalization estimator, ε the error (or loss) function, and τ is the target value required by the user. This means that the global surrogate model generation problem (i.e., finding the best approximation $s^* \in S$) for a given set of data points $D = (X, f(X))$ can be formally defined as

$$s^* = \arg\min_{t \in T} \arg\min_{\theta \in \Theta} \Lambda(\varepsilon, s_{t,\theta}, D) \tag{1}$$

such that

$$\Lambda(\varepsilon, s^*_{t,\theta}, D) \leqslant \tau$$

where $s_{t,\theta}$ is the parametrization θ (from a parameter space Θ) of s and $s_{t,\theta}$ is of model type t (from a set of model types T).

1. The terms surrogate model and metamodel are used interchangeably.

The first minimization over $t \in T$ is the task of selecting a suitable approximation model type, that is, a rational function, a neural network, a spline, etc. This is the model type selection problem. In practice, one typically considers only a single $t \in T$, though others may be included for comparison. Then given a particular approximation type t, the task is to find the hyperparameter assignment θ that minimizes the generalization estimator Λ (e.g., determine the optimal order of a polynomial model). This is the hyperparameter optimization problem, though generally both minimization's are simply referred to as the model selection problem. Many implementations of Λ have been described: the hold-out, bootstrap, cross validation, jack-knife, Akaike's Information Criterion (AIC), etc. Different criteria may also be combined in a multi-objective fashion. In this case ξ is really a matrix

$$\xi = \begin{bmatrix} \Lambda_1 & \varepsilon_1 & \tau_1 \\ \Lambda_2 & \varepsilon_2 & \tau_2 \\ \dots & \dots & \dots \\ \Lambda_m & \varepsilon_m & \tau_m \end{bmatrix}$$

with m the number of objectives. A simple example is minimizing the average relative cross validation error together with the maximum absolute deviation in the training points. An additional assumption is that f is expensive to compute. Thus the number of function evaluations $|f(X)|$ needs to be minimized and data points must be selected iteratively, at points where the information gain will be the greatest (Turner et al., 2007). Mathematically this means defining a sampling function

$$\phi(X_{j-1}) = X_j\,,\ j = 1,..,N$$

that constructs a data hierarchy

$$X_0 \subset X_1 \subset X_2 \subset \dots \subset X_N \subset X$$

of nested subsets of X, where N is the number of levels. X_0 is referred to as the *initial experimental design* and is constructed using one of the many algorithms available from the theory of Design and Analysis of Computer Experiments (Kleijnen et al., 2005). Once the initial design X_0 is available it can be used to seed the sampling function ϕ. An important requirement of ϕ is to minimize the number of sample points $|X_j| - |X_{j-1}|$ selected each iteration, yet maximize the information gain of each successive data level. Depending on the problem, ϕ can take into account different criteria (non-linearity of the response, smoothness/uncertainty of the model, location of the optima, etc.). This process is referred to as adaptive sampling, active learning, model updating, or sequential design.

An important consequence of the adaptive sampling procedure is that the task of finding the best approximation s^* (cfr. Equation 1) becomes a dynamic problem instead of a static one. Since the optimal model parameters will change as the amount and distribution of data points changes. This of course makes the problem more difficult.

3. Motivation

While the mathematical formulation of global surrogate modeling is clear cut, its practical implementation raises an obvious question: How should the minimization over $t \in T$ and $\theta \in \Theta$ in Equation 1 be performed? We discuss both cases in the following subsections.

3.1 Problem 1: Model Type Selection

The first minimization over $t \in T$ is the model type selection problem. Many model types exist: rational functions, Artificial Neural Networks (ANN), Support Vector Machines (SVM), Gaussian Process (GP) models, Multivariate Adaptive Regression Splines (MARS), Radial Basis Function (RBF) models, projection pursuit regression, rational functions, etc.

3.1.1 Background

From a theoretic standpoint, selecting the most suitable approximation method for a given response is a difficult problem that depends on the data characteristics (dimensionality, number of points, distribution, noise level, periodicity, etc) and the application constraints (accuracy, smoothness, ability to capture poles or discontinuities, execution speed, interpretability, extrapolation requirements, etc). Different application domains prefer different model types. For example rational functions are widely used by the electrical engineering community (Deschrijver and Dhaene, 2005) while ANN are preferred for hydrological modeling (Solomatine and Ostfeld, 2008). Differences in model type usage are often due to practical reasons. This is particularly true in industrial settings. For example: the designer adheres to common practice within his field, the final application restricts the designer to one particular type (e.g., rational models for EM systems), or the expertise is not available to properly try other methods.

Of course, this need not always be the case. The choice of the metamodel type can also be motivated by knowledge of the underlying physics[2] (Triverio et al., 2007) or by the special features the model provides: for example the uncertainty prediction based on random process assumption in Kriging methods[3] (Xiong et al., 2007). So remark there is no such thing as an inherently 'good' or 'bad' model. A model is only as good as the data it is based on and the expertise of the user that built it.

3.1.2 Classic Approach

If multiple model types are considered, the classic approach is to simply to try out different types and select the best one according to one or more accuracy criteria. There is ample literature available that benchmarks model types in this way (Simpson et al., 2001; Jin et al., 2001; Queipo et al., 2005; Yang et al., 2005; Chen et al., 2006; Wang and Shan, 2007; Chung and Alonso, 2000; Gano et al., 2006; Gu, 2001; Santner et al., 2003; Lim et al., 2007; Fang et al., 2005; Gorissen et al., 2009c). But claims that a particular model type is superior to others should always be met with some skepticism.

In order for the different benchmarking studies to be truly useful for a domain expert, the results of such studies must be collected and compiled into a general set of rules, recipe, or flowchart. To ease the discussion, let us denote such a compilation into a learning algorithm by L. L is then essentially a classifier that can predict which model type $t \in T$ to use based on data D and application requirements Γ:

2. Knowledge of the physics of the underlying system can make a particular model type to be preferred. For example, rational functions are popular for all kinds of Linear Time-Invariant systems since theory is available that can be used to prove that rational pole-residue models conserve certain physical quantities (Triverio et al., 2007).
3. Kriging models are closely related to GP models and and often Kriging and GP models are used as labels for the same techniques. Great similarities between GP models, SVM models, RBF models, and RBF Neural Networks exist as well, as has been discussed in Rasmussen and Williams (2006).

$$L(D,\Gamma) = t.$$

When executed L should then be able to give a specific recommendation as to which model type to use for a given problem. This recommendation should be more specific than the general rules of thumb that are available now. Experience shows this to be exactly what an application engineer wants. However, constructing such a learner L for any but the most restricted class of problems is a daunting undertaking for obvious practical reasons. Firstly, deciding which problem/application features to train the classifier on is far from trivial. Also even if this is done, the number of features can be expected to be high thus gathering the necessary data (by manually solving Equation 1) to train L accurately will be very computationally expensive.

Secondly, as mentioned above, the success of a model type largely depends on the expertise of the user, the quality of the data, and even the quality of the software implementation of the technique. Neural networks are a good example in this respect. In the right hands they are able to perform very well on many problems. However, if poor choices are made with regard to training function, topology selection, generalization control, training parameters, software library, etc. they may seem to perform poorly. How to take into account this information in L?

Thirdly, a more fundamental problem with this approach is that data must be available in order for the reasoner to work. However, if only a simulation code is available (as is often the case) data must be collected, and the optimal data collection strategy that minimizes the number of points depends on the model type. Also, the optimal model type will change depending on how much data is available. One could argue to instead train L only on the data characteristics which are known in advance (e.g., dimensionality, noise level, etc.). The question is then again, which characteristics are most important? Furthermore, in many cases not much is known about the true behavior of the response thus there will typically not be enough information to train L accurately.

This brings us to the final point. A main reason for turning towards global surrogate modeling methods is that little is known about the behavior of the response (Simpson et al., 2008). The goal is to get insight into that behavior in a computationally cheap way by applying surrogate methods. Another reason why information about the data may be scarce is that the source of the data is confidential or proprietary and very little information is disclosed. In these situations using or training L becomes very difficult.

Finally, we must stress that we do ***not*** say that this problem is too difficult and not worth trying to solve. Indeed many such problems exist and are currently being tackled, particularly in medicine. Instead we argue that users of global surrogate modeling methods can benefit from a more dynamic approach that is flexible, can be easily applied to a wide range of different problems, can easily incorporate new fitting techniques and process knowledge, and naturally integrates with an adaptive data collection procedure. We shall revisit this point in sections 3.3 and 6.

3.2 Problem 2: Model Parameter Selection

Assuming the model type selection problem has been solved, there remains the model parameter selection problem (the minimization over $\theta \in \Theta$ in Equation 1). For example, finding the optimal C, ε and σ parameters in the case of RBF SVMs. This is the classic hyperparameter optimization problem that also depends on the data characteristics (for example the optimal correlation function and correlations parameters of a Kriging model will depend on the data distribution (Gorissen et al., 2008b; Toal et al., 2008). Some models are more sensitive to changes in their parameters than others and usually it takes a great deal of experience to know how all parameters should be set. Sometimes this problem is solved through trial and error, but usually it is tackled as an optimization problem and classic optimization algorithms are used guided by a performance metric.

A huge amount of research has been done on this topic, particularly in the machine learning community (see Section 4). This particular problem is not the main focus of this work. Rather we are more interested in tackling the first problem.

3.3 Proposed Solution

While we are primarily interested in the first problem, the approach described in this paper naturally incorporates problem 2 as well. In both cases there is little theory that can be used as a guide. It is in this setting that the evolutionary approach can be expected to do well. We describe the application of a single GA with speciation to both problems: the selection of the surrogate type and the optimization of the surrogate model parameters (= hyperparameter optimization). In addition, we do not assume all data is available at once but must be sampled incrementally since it is expensive (active learning).

The idea is to maintain a heterogeneous population of surrogate model types and let them evolve cooperatively and dynamically with the changing data distribution. The details will be presented in Section 5 and a critique in Section 6. In addition, an implementation in the form of a Matlab toolbox is available for download from `http://www.sumo.intec.ugent.be`.

4. Related Work

The evolutionary generation of regression models for given input-output data has been widely studied in the genetic programming community (Vladislavleva et al., 2009; Streeter and Becker, 2003; Yeun et al., 2004). Given a set of mathematical primitives (+*, sin, exp, /, x, y,* etc.) the space of symbolic expression trees is searched to find the best function approximation. The application of GAs to the optimization of model parameters of a single model type (homogeneous evolution) has also been common (Chen et al., 2004; Lessmann et al., 2006; Tomioka et al., 2007; Friedrichs and Igel, 2005; Zhang et al., 2000) and the extensive work by Yao (1999); Yao and Xu (2006). Integration with adaptive sampling has also been discussed (Busby et al., 2007). However, these efforts do not tackle the model *type* selection problem, they restrict themselves to a particular method (e.g., SVMs or neural networks). As Knowles and Nakayama (2008) state "*Little is known about which types of model accord best with particular features of a landscape and, in any case, very little may be known to guide this choice.*". Likewise, Solomatine and Ostfeld (2008) note: "*...it is important to stress that there are always situations when one model type cannot be applied or suffers from inadequacies and can be well complemented or replaced by another one*". Thus an algorithm to help solve this problem in a dynamic, automated way is very useful (Keys et al., 2007). This is also noticed by Voutchkov and Keane (2006) who compare different surrogate models for approximating each objective during optimization. They note that in theory their approach allows the use of a different model type for each objective. However, such an approach will still require an a priori model type selection and does not allow for dynamic switching of the model type or the generation of hybrids.

There has also been much research on the use of surrogate models in evolutionary optimization of expensive simulators (to approximate the fitness function). References include Jin et al. (2002), Regis and Shoemaker (2004), Paenke et al. (2006) and Emmerich et al. (2006), the work by Ong et al. (2006), and more recently by Lim et al. (2007). In general the theory is referred to as Surrogate

Based Optimization or Metamodel Assisted Optimization. A good overview reference is given by Eldred and Dunlavy (2006) and Queipo et al. (2005). For example, Lim et al. (2007) compare the utility of different local surrogate modeling techniques (quadratic polynomials, GP, RBF, ...) including the use of (fixed) ensembles, for optimization of computationally expensive simulation codes. Local surrogates are used together with a trust region framework to quickly and robustly identify the optimum. As noted in the introduction, the contrast with this work is that references such as Lim et al. (2007) are interested in the optimum and not the surrogate itself (they make only a "*mild assumption on the accuracy of the metamodeling technique*"). In addition the model parameters are taken as fixed and there is no integration with active learning. In contrast we place very strong emphasis on the surrogate model accuracy, the automatic setting of the hyperparameters, and the efficient sampling of the *complete* design space.

The work of Sanchez et al. (2006) and Goel et al. (2007) is more useful in our context since they provide new algorithms for generating an optimal set of ensemble members for a fixed set of data points (no sampling). Unfortunately, though, the parameters of the models involved must still be chosen manually. Nevertheless, their approaches are interesting, and can be used to further enhance the approach presented here. For example, instead of returning the single final best model, an optimal ensemble member selection algorithm can be used to return a potentially much better model based on the final population or Pareto front.

From machine learning the work in B. et al. (2004) is also related. The authors describe an interesting classification algorithm *COMB* that combines online an ensemble of active learners so as to expedite the learning progress in pool-based active learning. In their terminology an active learner is a combination of a model type and a sampling algorithm. A weighted ensemble of active learners is maintained and each learner is allowed to express interest in a pool of unlabeled training points. Depending on the interests of the active learners, an unlabeled point is selected, labeled by the teacher, and based on the added value of that point the different active learners are punished or rewarded. Internally the active learners are SVM models whose parameters are chosen manually. In principle, with a number of approximations one could adapt the algorithm to the regression case. If one then also included hyperparameter optimization, the result would be very similar to the SUMO-Toolbox (cfr. Section 5.2) configured with one or more of the *Error*, LRM, or EGO (Jones et al., 1998) sample selection algorithms, but without the ability to combine different criteria. However, a problem would be that *COMB* assumes a pool of unlabeled training data is given. However, when modeling a simulation code in regression no such pool is available. Some external algorithm would still be needed to generate it in order for *COMB* to work. *COMB* does also naturally allow for different model types but in a more static way than the algorithm in Section 5.3: there is no hyperparameter optimization, the number of each active learning type remains fixed (though the weights can change) leading to a potentially high computational cost, and hybrid models are not considered. The extension to the multi-objective case is also non-trivial. Of course *COMB* could be extended to incorporate such features, but the result would be very similar to the work presented here. Nevertheless, the specific scoring functions, probability weightings, and ensemble weight updates, seem very useful and could be implemented in the SUMO-Toolbox to complement the approach presented here.

Finally, the work by Escalante et al. (2008) is most similar to the topic of this paper. Escalante et al. (2008) consider the problem of finding the optimal classifier and associated hyperparameters for a given classification problem (active learning is not considered). A solution is encoded as a vector and Particle Swarm Optimization (PSO) is used to search for good classifiers. Good results are shown on various benchmarks. Unlike the GA approach, however, it is less straightforward to cater for multiple sub-populations, giving models room to mature independently before entering competition. The use of operators tuned to specific models is also difficult (to increase the search efficiency). In effect, the PSO approach takes a top-down view, using a high level encoding in a high dimensional space, a typical particle has 25 dimensions (Escalante et al., 2007). In contrast the GA approach is bottom up, the model specific operators result in a much smaller search space, different for each method (e.g., 1 for the spline models and 2 for the SVM models). This leads to a more efficient search requiring less fitness evaluations and facilitates the incorporation of prior knowledge. In addition, by using PSO there is no natural way of enabling hybrid solutions (ensembles) without extending the encoding and further increasing the search space. In contrast, the hybrid solutions arise very naturally in the GA framework and do not impact the search space of the other model types. The same is true of the extension to the multi-objective case, a very natural step in the GA case.

In sum, in by far the majority of the related work considered by the authors, speciation was always constrained to one particular model type, for example neural networks in Stanley and Miikkulainen (2002). The model type selection problem was still left as an a-priori choice for the user. Or, if multiple model types are used, the hyperparameters are typically kept fixed and there is no tie-in with the active learning process.

5. Heterogeneous Evolution of Surrogate Models

This Section discusses how different surrogate models may be evolved cooperatively in order perform model type selection.

5.1 Speciated Evolution

Since GAs are population-based they easily lend themselves to parallelism. The terms Parallel Genetic Algorithms (PGA) or Distributed Genetic Algorithms (DGA) refer to the case whenever the population is divided up in some way, be it to improve the computational efficiency or search efficiency. Unfortunately though, the terminology varies between authors and can be confusing (Nowostawski and Poli, 1999; Alba and Tomassini, 2002). From a biological standpoint it makes sense to consider *speciation:* genomes that differ considerably from the rest of the population are segregated and continue to evolve semi-independently, forming a new species.

The *island model* (Whitley et al., 1999.; Hocaoglu and Sanderson, 2001; Giannakoglou et al., 2006) is probably the most well known PGA. Different sub-populations, called *demes*, exist (initialized differently) and sporadic migration can occur between islands allowing for the exchange of genetic material between species and inter-species competition for resources. Selection and recombination are restricted per deme, such that each sub-population may evolve towards different locally optimal regions of the search space (also called *niches*). An advantage of using migration is that it allows sub-species to mature in semi-isolation without being forced to consistently engage in competition. This is particularly useful for the application of this paper. The island model introduces five new parameters: the migration topology, the migration frequency, the number of individuals to migrate, a strategy to select the emigrants, and a replacement strategy to incorporate the immigrants. The island model is illustrated in Figure 1 for two topologies.

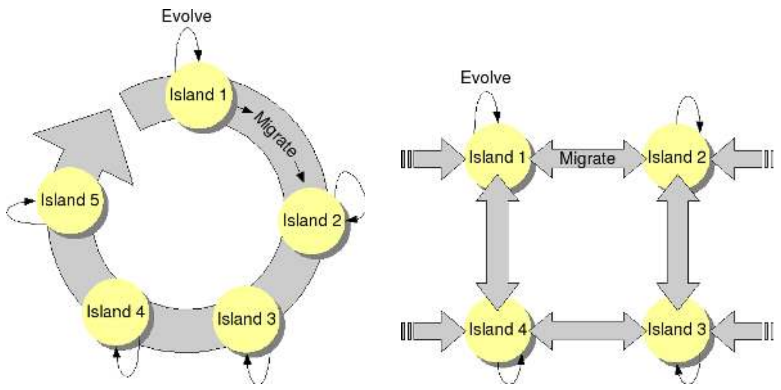


Figure 1: Ring (left) and grid (right) migration topologies in the Island Model

5.2 Global Surrogate Modeling Control Flow

Before we can discuss the concrete implementation of the automatic model type selection algorithm it is important to revisit the general global surrogate modeling methodology described in Section 2. It is important to understand the general control flow since it forms the basis for the evolutionary algorithm described in the next Section.

The general methodology is as follows: Initially, a small initial set of samples is chosen according to some experimental design (e.g., Latin hypercube, Box-Behnken, etc.). Based on this initial set, one or more surrogate models are constructed and their hyperparameters optimized according to a chosen hyperparameter optimization algorithm (e.g., BFGS, Particle Swarm Optimization (PSO), Genetic Algorithm (GA), DIRECT, NSGA-II, etc.). Models are assigned a score based on one or more measures (e.g., cross validation, Akaike's Information Criterion (AIC), etc.) and the optimization continues until no further improvement is possible. The models are then ranked according to their score and new samples are selected based on the best performing models and the behavior of the response (the exact criteria depend on the active learning algorithm used). The hyperparameter optimization process is continued or restarted intelligently and the whole process repeats itself until one of the following three conditions is satisfied: (1) the maximum number of samples has been reached, (2) the maximum allowed time has been exceeded, or (3) the user required accuracy has been met.

Recall that the adaptive sampling procedure has an important effect on the hyperparameter optimization. The non-stationary data distribution makes the model parameter optimization surface dynamic instead of static (as is typically assumed).

A readily available implementation of the control flow described in this Section, and the one we shall use for the experiments in this paper, is available as the **SU**rrogate **MO**deling Toolbox (SUMO Toolbox) (Gorissen et al., 2009c) from `http://www.sumo.intec.ugent.be`.

5.3 Algorithm

Algorithm 1 Automatic global surrogate modeling with heterogeneous evolution and active learning

```
01. $X_0 = initialExperimentalDesign()$ ;
02. $X = X_0$;
03 . $f|_X = evaluateSamples(X)$ ;
04 .$T = \{t_1, ..., t_h\}$;
05. $M_i = \text{createInitialModels}(t_i, popsize_i)$; $i = 1,..,h$
06. $M = \bigcup_{i=1}^{h} M_i$;

07 . while ( $\xi$  not reached)  do
08.   $scores = \{\}$;   $gen = 1$;
09 .   while ( $termination\_criteria$  not reached)  do
10.     foreach$M_i \subseteq M$  do
11.       $scores_i = \text{fitness}(M_i, X, f|_X, \xi)$;
12.       $elite = \text{sort}([scores_i; M_i])|_{1:el}$;
13.       $parents = \text{select}(scores_i, M_i)$;
14.       $parents_{xo} = \text{selectXOParents}(parents, p_c)$;
15.       $offspring_{xo} = \text{crossover}(parents_{xo}, ES_{diff}, ES_{max})$;
16.       $parents_{mut} = parents \backslash parents_{xo}$;
17.       $offspring_{mut} = \text{mutate}(parents_{mut})$;
18.       $M_i = elite \bigcup offspring_{mut} \bigcup offspring_{xo}$;
19.       $scores = scores_i \cup scores$;
20.     end
21.     if $(mod(gen, m_i) = 0)$
22.       $M = \text{migrate}(M, scores, m_f, m_d)$
23.     end
24.     $M = \text{extinctionPrevention}(M, T_{min})$;
25.     $gen = gen + 1$;
26.   end
27 .   $X_{new} = \text{selectSamples}(X, f|_X, M)$;
28 .   $f|_{X_{new}} = \text{evaluateSamples}(X_{new})$;
29 .   $[X, f|_X] = merge(X, f|_X, X_{new}, f|_{X_{new}})$ ;
30 . end

31 .  returnbestModel($M$);
```

We now present the concrete GA for heterogeneous evolution as it is embedded (as a plugin) in the SUMO Toolbox. The speciation model used is the island model since we found it the most natural way of evolving multiple model types while still allowing for hybrid solutions. The algorithm is based on the Matlab GADS toolbox and works as follows (see Algorithm 1 and reference (Gorissen, 2007) for more details): After the initial DOE has been calculated (cfr. the control flow in Section 5.2), an initial sub-population M_i is created for each model type $t \in T$ $(i = 1,..,h)$. The exact creation algorithm is different for each model type so that model specific knowledge can be exploited.

Subsequently, each deme is allowed to evolve according to an elitist GA. Parents are selected according a selection algorithm (e.g., tournament selection) and offspring undergo either crossover (with probability p_c) or mutation (with probability $1 - p_c$). The models M_i are implemented as Matlab objects (with full polymorphism) thus each model type can choose its own representation and mutation/crossover implementations (this implements the minimization over $\theta \in \Theta$ of Equation 1). While mutation is straightforward, the crossover operator is more involved (see Section 5.5 below).

The fitness function calculates the quality of the model fit, according to criteria ξ. The current deme population is then replaced with its offspring together with *el* elite individuals. Once every deme has gone through a generation, migration between individuals is allowed to occur at migration interval m_i, with migration fraction m_f and migration direction m_d (a ring topology is used). The migration strategy is as follows: the $l = (|M_i| \cdot m_f)$ fittest individuals of M_i replace the l worst individuals in the next deme (defined by m_d). As in Pei and Goodman (2001), migrants are duplicated, not removed from the source population. Note that in this contribution we are primarily concerned with inter-model speciation (speciation as in different model types). Intra-model speciation (e.g., through the use of fitness sharing within one model type) is something which was not done but could easily be incorporated.

Once the GA has terminated, control passes back to the main global surrogate modeling algorithm of the SUMO Toolbox. At that point M contains the best set of models that can be constructed for the given data. If the accuracy of the models is sufficient the main loop terminates. If not, a new set of maximally informative sample points is selected based on several criteria (quality of the models, non-linearity of the response, etc.) and scheduled for evaluation. Once new simulations become available the GA is resumed.

Note that sample evaluation and model construction/hyperparameter optimization run in parallel. For clarity, algorithm 1 shows them running sequentially but this is not what happens in practice. In reality both are interleaved to allow an optimal use of computational resources.

5.4 Extinction Prevention

Initial versions of this algorithm exposed a major shortcoming, specifically due to the fact that models are being evolved. Since not all data is available at once but trickles in, $|X_j| - |X_{j-1}|$ samples at a time, models that need a reasonable-to-large number of samples to work well will be at a huge disadvantage initially. Since they perform badly at first, they may get overwhelmed by other models who are less sensitive to this problem. In the extreme case where they are driven extinct, they will never have had a fair chance to compete when sufficient data *does* become available. They may even have been the superior choice had they still been around.[4] Therefore an Extinction Prevention (EP) algorithm was introduced that ensures a model type can never disappear completely.

EP works by monitoring the population and each generation recording the number of individuals of each model type. If this number falls below a certain threshold T_{min} for a certain model type, the EP algorithm steps in and ensures the model type has its numbers replenished up to the threshold. This is done by re-inserting the last models that disappeared for that type (making copies if necessary). The re-inserted models replace the worst individuals of the other model types (who do have sufficient numbers) evenly.

Strictly speaking, EP goes completely against the survival of the fittest principle in evolutionary algorithms. By using it we are manually working against selection, preserving model types which

4. As an example, this observation was often made when using rational models on electro-magnetic data.

give poor results at that point in time. However, in this setting it seems a fair measure to take (we do not want to risk loosing a model type completely) and improves results in most cases (*see* Section 10). At the same time it is straightforward to implement and understand, needing no special control parameters. All it has to ensure is that a species is never driven extinct.

5.5 Heterogeneous Recombination

The attentive reader will have noticed that one major problem remains with the implementation as discussed so far. The problem lies in the genetic operators, more specifically in the crossover operator. Migration between demes means that model types will mix. This means that a set of parents selected for reproduction may contain more than one model type. The question then arises: how to perform recombination between two models of completely different types. For example, how to meaningfully cross an Artificial Neural Network with a rational function? The solution we propose here is to use ensembles (behavioral recombination). If two models of different types are selected to recombine, an ensemble is created with the models as ensemble members. Thus, as soon as migration occurs, model types start mixing, and ensemble models arise as a result. These are treated as a distinct model type just as the other model types.

However, the danger with this approach is that the population may quickly be overwhelmed by large ensembles containing duplicates of the best models (as was noticed during initial tests). To counter this phenomenon we apply the similarity idea from Holland's sharing concept (Holland, 1975). Individual models will try to mate only with individuals of the same type. Only in the case where selection has made this impossible shall different model types combine to form an ensemble. In addition we enforce a maximum ensemble size ES_{max} and require that ensemble members must differ ES_{diff} percent in their response (their 'behavior'). This is calculated by evaluating the models on a dense grid.

This leaves us with only three cases left to explain:

1. *ensemble - ensemble* recombination: a single-point crossover is made between the ensemble member lists of each model (note that the type of the ensemble members is irrelevant)

2. *ensemble - model* recombination: the model replaces a randomly selected ensemble member with probability p_{swap} or gets absorbed into the ensemble with probability $1 - p_{swap}$ (respecting ES_{max} and ES_{diff}).

3. *ensemble* mutation: one ensemble member is randomly deleted

Besides enabling hybrid solutions, using ensembles has the additional benefit of allowing a model to lie 'dormant' in an ensemble with the possibility of re-emerging later (e.g., if after mutation only one ensemble member remains). Note that, in contrast to Lim et al. (2007) for example, the type of the ensemble members is not fixed in any way but varies dynamically.

We have not yet mentioned what type of ensemble will be used. There are several methods for combining the outputs of models, such as average, weighted average, Dempster-Shafer methods, using rank-based information, supra-Bayesian approach, stacked generalization, etc (Sharkey, 1996). To keep the implementation straightforward and the complexity (number of parameters) low we have opted for a simple average ensemble. Of course different, more powerful combination methods could be used instead and they will only improve results. The exact method used is of lesser importance since it does not change the methodology. The advantage of a simple average ensemble is that it works in all cases: It makes no assumption on the model types involved, nor does it mandate any changes to the models or training algorithms (for example, like negative correlation learning) since this is not always possible (e.g., when using proprietary, application specific, modeling code).

5.6 Multi-objective Model Selection

A crucial aspect of the model generation algorithm is the choice of a suitable criteria ξ. In practice it turns out that selecting an appropriate function for Λ, ε and a target value for τ is difficult. Particularly if little is known about the structure of the response. This is related to the "*The 5 percent problem*" (Gorissen et al., 2009b). The fundamental reason for this difficulty is that an approximation task inherently involves multiple, conflicting, criteria (Li and Zhao, 2006). Thus a multi-objective approach is very useful here since it enables the use of multiple criteria during the hyperparameter optimization process (see Jin and Sendhoff 2008 for an excellent overview of this line of research).

Secondly, it is not uncommon that a simulation engine has multiple outputs that all need to be modeled (Conti and O'Hagan, 2007). The direct approach is to model each output independently with separate models (possibly sharing the same data). This, however, leaves no room for trade-offs nor gives any information about the correlation between different outputs. Instead of performing two modeling runs (doing a separate hyperparameter optimization for each output) both outputs can be modeled simultaneously if models with multiple outputs are used in conjunction with a multi-objective optimization routine.

In both cases such a multi-objective approach can be integrated with the automatic surrogate model type selection algorithm described here. This means that the best model type can vary per criteria or, more interestingly, that it enables automatic selection of the best model type for each output without having to resort to multiple runs. A full discussion of these topics is out of scope for this paper. However, details and some initial results can already be found in Gorissen et al. (2009a) and Gorissen et al. (2009b).

6. Critique

The algorithm presented so far has a number of strengths and weaknesses. The obvious advantage is the ability to perform automatic selection of the model type and complexity for a given data source (no need to do multiple parallel runs or train a complex classifier). In addition the algorithm is generic in that it is independent of the data origin (application), model type, and data collection strategy. New approximation methods can easily be incorporated without changing the algorithm. Problem specific knowledge and model type specific optimizations based on expert knowledge can also be incorporated if needed (i.e., by customizing the genetic operators). Furthermore, the algorithm naturally integrates with the data collection strategy, allowing the best model type to change dynamically and naturally allows for hybrid solutions. Finally, it naturally extends to the multi-objective case (Section 5.6) and can be easily parallelized to allow for faster computations (though the computational cost is still outweighed by the simulation cost).

The main disadvantage is due to the fact that the approach is based on evolutionary algorithms: full determinism can not be guaranteed. This raises the obvious question of how stable the convergence is over multiple runs. The same can be said of standard approaches towards hyperparameter optimization (which typically include randomization) or for any algorithm involving a GA for that matter. Formulating theoretical foundations in order to come to convergence guarantees for GAs is a difficult undertaking and has been the topic of intense research ever since their inception in the late 80s. Characterizing the performance of genetic algorithms is complex and depends on the application domain as well as the implementation parameters (Rawlins, 1991). Most theoretic work has been done on schema theorems for the Canonical Genetic Algorithm (CGA), which try to prove convergence in a simplified framework using a binary representation. However prediction of the future behavior of a GA turns out to be very difficult and much controversy remains over the usefulness of these theorems (Poli, 2001; Goldberg, 1989). Theoretical work on other classes of GAs or using specific operators has also been done (Nakama, 2008; Neubauer, 1997; Rawlins, 1991; Qi and Palmieri, 1994a,b) but is unfortunately of little practical use here. For example, the work in Ankenbrandt (1990) requires the calculation of fitness ratios, but this is impractical (and computationally expensive) to do in this situation and the results will vary with the application.

Thus, for the purposes of this paper a full mathematical treatment of algorithm 1 and its convergence is out of scope. Due to the island model, sampling procedure, and heterogeneous representation/operators used, such a treatment will be far from trivial to construct and distract from the main theme of the paper. In addition its practical usefulness would remain questionable due to the many assumptions that will be required. However, gaining a deeper theoretical insight into the robustness of the algorithm is still very important. A sensitivity study of the main GA parameters involved will shed more light on this issue.

Theoretical remarks aside, the authors have found that in practice the approach works quite well. If reasonable population sizes are used together with migration and the extinction prevention algorithm described in Section 5.4, the results of the algorithm are quite robust and give useful results and insights into the modeling problem. Besides the results given in this paper, good results have also been reported on various real world problems from aerodynamics (Gorissen et al., 2009a), electronics (Gorissen et al., 2008a), hydrology (Couckuyt et al., 2009), and chemistry (Gorissen, 2007).

In sum, this approach is useful if: little information is known about the expected structure of the response, if it is unclear which model type is most suited to the problem, data is expensive and must be collected iteratively, and hybrid solutions are useful. In other cases, for example it is clear from a priori knowledge which model type will be the most suitable (e.g., based on existing rules of thumb for a well defined, restricted problem), this approach should not be applied, save as a comparison.

7. Test Problems

We now consider five test problems to which we apply the heterogeneous GA (from now on abbreviated by HGA). The objective is to validate if the best model type can indeed be determined automatically, and in a way that is cheaper and better than the simple brute force method: doing multiple, single model type runs in parallel. The problems include 2 predefined mathematical functions, and real-world problems from electronics and aerodynamics.

The dimensionality of the examples ranges from 2 to 13. This is no inherent limit but simply depends on the model types used. For example if only SVM-type models are used the number of dimensions can be arbitrarily high, while for smoothing spline models the dimensionality should be kept low. It all depends on which model types make up the population.

We also hope to see evidence of a ‘battle’ between model types. While initially one species may have the upper hand, as more data becomes available (dynamically changing hyperparameter optimization landscape) a different species may become dominant. This should result in clearly noticeable population dynamics, a kind of oscillatory stage before convergence. We briefly discuss each of the test problems in turn.

7.1 Ackley Function (AF)

The first test problem is Ackley’s Path, a well known benchmark problem from optimization. Its mathematical definition for d dimensions is:

$$\begin{aligned} F(\vec{x}) &= -20 \cdot \exp\left(-0.2\sqrt{\frac{1}{d} \cdot \sum_{i=1}^{d} x_i^2}\right) \\ &\quad - \exp\left(\frac{1}{d} \cdot \sum_{i=1}^{d} \cos(2\pi \cdot x_i)\right) + 20 + e \end{aligned}$$

with $x_i \in [-2,2]$ for $i = 1,...,d$. For easy visualization we take $d = 2$. For this function a validation set and a test set of 5000 random points each is available. Although this is a function from optimization we are not interested in optimizing it, rather in reproducing it using a regression method with minimal data.

7.2 Kotanchek Function (KF)

The second predefined function is the Kotanchek function (Smits and Kotanchek, 2004). Its mathematical definition is given as:

$$F(x_1, x_2, u_1, u_2, u_3) = \frac{e^{-x_2^2}}{1.2 + x_1^2} + \varepsilon$$

with $x_1 \in [-2.5, 1.5]$, $x_2 \in [-1.0, 3.0]$, and with ε uniform random simulated numeric noise with mean 0 and variance 10^{-4}. As you can see only the first two variables are relevant. For this function a validation set and a test set of 5000 scattered points each is available.

7.3 EM Example (EE)

The fourth example is a 3D Electro-Magnetic (EM) simulator problem (Lehmensiek, 2001). Two perfectly conducting round posts, centered in the E-plane of a rectangular waveguide, are modeled, as shown in Figure 2. The 3 inputs to the simulation code are: the signal frequency f, the diameter of the posts d, and the distance between the two posts w. The outputs are the complex reflection and transmission coefficients S_{11} and S_{21}. The simulation model was constructed for a standard WR90 rectangular waveguide with $f \in$[7 GHz, 13 GHz], $d \in$[1 mm, 5 mm] and $w \in$[4 mm, 18 mm]. In addition, a 25^3 data set is available for testing purposes.

7.4 LGBB Example (LE)

NASA’s Langley Research Center is developing a small launch vehicle (SLV) (Pamadi et al., 2004; Rogers et al., 2003) that can be used for rapid deployment of small payloads to low earth orbit

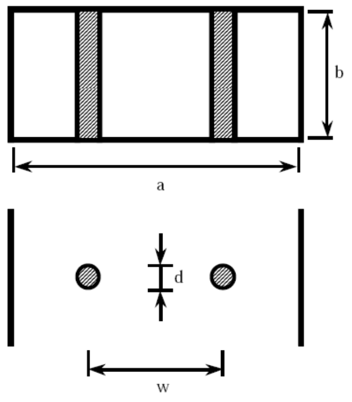
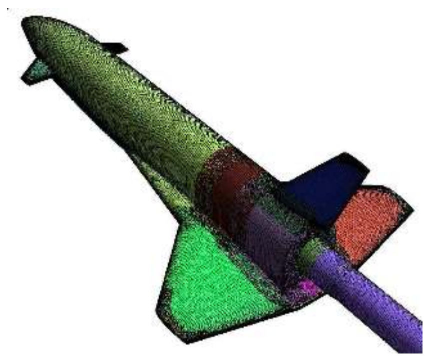
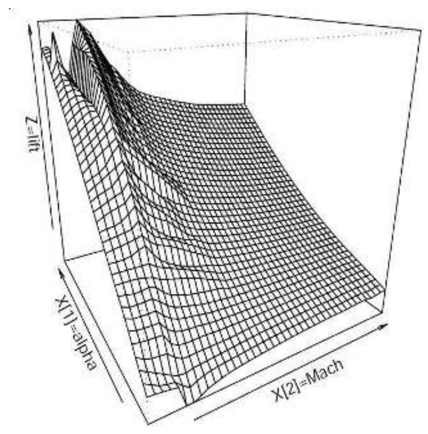


Figure 2: Cross sectional view and top view of the inductive posts (Lehmensiek, 2001)



(a) LGBB geometry (Rogers et al., 2003)



(b) Lift plotted as a function of speed and angle of attack with side-slip angle fixed to zero (Gramacy et al., 2004).

Figure 3: LGBB Example

at significantly lower launch costs, improved reliability and maintainability. The vehicle is a three-stage system with a reusable first stage and expendable upper stages. The reusable first stage booster, which glides back to launch site after staging around Mach 3 is named the Langley Glide-Back Booster (LGBB). In particular, NASA is interested in the aerodynamic characteristics of the LGBB from subsonic to supersonic speeds when the vehicle reenters the atmosphere during its gliding phase.

More concretely, the goal is to gain insight about the response in lift, drag, pitch, side-force, yaw, and roll of the LGBB as a function of three inputs: Mach number, angle of attack, and side slip angle. For each of these input configurations the Cart3D flow solver is used to solve the inviscid Euler equations over an unstructured mesh of 1.4 million cells. Each run of the Euler solver takes on the order of 5-20 hours on a high end workstation (Rogers et al., 2003). The geometry of the LGBB used in the experiments is shown in Figure 3a.

Figure 3b shows the lift response plotted as a function of speed (Mach) and angle of attack (alpha) with the side-slip angle (beta) fixed at zero. The ridge at Mach 1 separates subsonic from supersonic cases. From the figure it can be seen there is a marked phase transition between flows at subsonic and supersonic speeds. This transition is distinctly non-linear and may even be non-differentiable or non-continuous (Gramacy et al., 2004). Given the computational cost of the CFD solvers, the LGBB example is an ideal application for metamodeling techniques. Unfortunately access to the original simulation code is restricted. Instead a data set of 780 points chosen adaptively according to the method described in Gramacy et al. (2004) was used.

7.5 Boston Housing Example (BH)

The Boston Housing data set contains census information for 506 housing tracts in the Boston area and is a classic data set used in statistical analysis. It was collected by Harrison et al. and described in Harrison and Rubinfeld (1978). In the case of regression the objective is to predict the Median value of owner-occupied homes (in $1000's) from 13 input variables (e.g., per capita crime rate by town, nitric oxide concentration, pupil-teacher ratio by town, etc.).

8. Model Types

For the tests the following model types are used: Artificial Neural Networks (ANN), rational functions, RBF models, Kriging models, LS-SVMs, and for the AF example: also smoothing splines. For the EM example only the model types that support complex valued outputs directly (rational functions, RBF, Kriging) were included. Each type has its own representation and genetic operator implementation (thanks to the polymorphism as a result of the object oriented design). As stated in subsection 5.5 the result of a heterogeneous recombination will be an averaged ensemble. So in total up to seven model types will be competing to approximate the data. Remember that all model parameters are chosen automatically as part of the GA. No user input is required, the models and data points are generated automatically.

The ANN models are based on the Matlab Neural Network Toolbox and are trained with Levenberg Marquard backpropagation with Bayesian regularization (MacKay; Foresee and Hagan, 1997) (300 epochs). The topology and initial weights are determined by the GA. When run alone (without the HGA) this results in high quality models with a much faster run time than training the weights by evolution as well. Nevertheless, the high level Matlab code and complex training function do make the ANNs much slower than any of the other model types.

The LS-SVM models are based on the implementation from Suykens et al. (2002), the kernel type is fixed to RBF, leaving c and σ to be chosen by the GA. The Kriging model implementation is based on Lophaven et al. (2002) (except for the EM example) and the correlation parameters are set by the GA (the regression function is set to linear and the correlation function to Gaussian). The RBF models (and the Kriging models for the EM example, since the data is complex valued) are based on a custom implementation where the regression function, correlation function, and correlation parameters are all evolved. The rational functions are also based on a custom implementation, the free parameters being the orders of the two polynomials, the weights of each parameter, and which parameters belong in the denominator. The spline models are based on the Matlab Splines Toolbox and only have one free parameter: the smoothness.

Remember that the specific model types chosen for the different tests is less important. This can be freely chosen by the user. What is important is rather how these different model types are used together in a single algorithm. Thus a full explanation of the virtues of each model types, as well as the representation and genetic operators used is out of scope for this paper and would consume too much space. Details can be found in Gorissen (2007) or in the implementation that is available as part of the SUMO Toolbox.

9. Experimental Setup

The following subsections describe the configuration settings used (and their motivation) for performing the experiments.

9.1 Sample Selection Settings

For the LE and BH examples only a fixed, small size, data set is available. Thus, selecting samples adaptively makes little sense. So for these examples the adaptive sampling loop was switched off. For the other examples the settings were as follows: an initial optimized Latin hypercube design, using the method from Ye et al. (2000), of size 50 is used augmented with the corner points. Modeling is allowed to commence once at least 90% of the initial samples are available. Each iteration a maximum of 50 new samples are selected using the Local Linear (LOLA) adaptive sampling algorithm (Crombecq et al., 2009). LOLA identifies new sample locations by making a tradeoff between eploration (covering the design space evenly) and exploitation (concentrating on regions where true response is nonlinear). LOLA's strengths are that it scales well with the number of dimensions, makes no assumptions about the underlying problem or surrogate model type, and works in both the $\mathbb{R}$ and $\mathbb{C}$ domains. LOLA is able to automatically identify non-linear regions in the domain and sample these more densely compared to more linear, 'flatter' regions.

By default LOLA does not rely on the (possibly misleading) approximation model, but only on the true response. This is useful here since it allows us to consider the model selection results independent from the sample selection settings. I.e., the final distribution of points chosen by LOLA is the same across all runs and model types. This means that any difference in performance between models can not be due to differences in sample distribution. However, in many cases it may be desirable to also include information about the surrogate model itself when choosing potential sample locations. In this case the LOLA algorithm can be combined with one or more other sampling criteria that do depend on model characteristics (for example the *Error*, LRM, and EGO algorithms available in SUMO).

9.2 GA Settings

The GA is run for a maximum of 15 generations between each sampling iteration (after sampling, the GA continues with the final population of the previous iteration). It terminates if one of the following conditions is satisfied: (1) the maximum number of generations is reached, or (2) 8 generations with no improvement. The size of each deme is set to 15. The migration interval m_i is set to 7, the migration fraction m_f to 0.1 and the migration direction is *both* (copies of the m_f best individuals from island i replace the worst individuals in islands $i-1$ *and* $i+1$). A stochastic uniform selection function was used. Since we want to find the best approximation over the *complete* design space, the fitness of an individual is defined as the root relative square error (RRSE):

$$RRSE(y_i, \tilde{y_i}) = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \tilde{y_i})^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

where $y_i, \tilde{y_i}, \bar{y}$ are the true, predicted, and mean true response values respectively. Intuitively the *RRSE* indicates how much better an approximation is than the most simple approximation possible (the mean) (Ganser et al., 2007). In the case of the BH and LE examples no separate validation set is available, instead 20% of the available data is reserved for this purpose (taking care to ensure the validation set is representative of the full data set by maximizing the minimum distance between validation points). Note that we are using a validation set since it is cheap and we have enough data available. In the case where data is scarce we would most likely use the more expensive *k*-fold cross validation as a fitness measure. This is the case for the EM example.

The remaining parameters are set as follows: $p_m = 0.2, p_c = 0.7, k = 1, p_{swap} = 0.8, el = 1, ES_{max} = 3, ES_{diff} = 0.1, T_{min} = 2$. The random generator seed was set to Matlab's default initial seed.

9.3 Termination Criteria

In case of adaptive modeling only (no sample selection), the objective is to see what the most accurate model is that can be found in a limited period of time (= a typical use case). Thus the required accuracy (= target fitness value) is set to 0. For the LE the timeout is set to 180 minutes. For the BH example the timeout is significantly extended to 1200 minutes. Given the high dimensionality, the noise and discontinuities in the input domain it is a hard problem to fit accurately. In this case we are more interested to see how the population would evolve over such an extended period of time.

In case of adaptive sampling, the criteria are: a target accuracy (RRSE) of 0.01, and for the AF example a maximum number of 500 data points is enforced (to see what performance can be reached with a limited sample budget).

9.4 Others

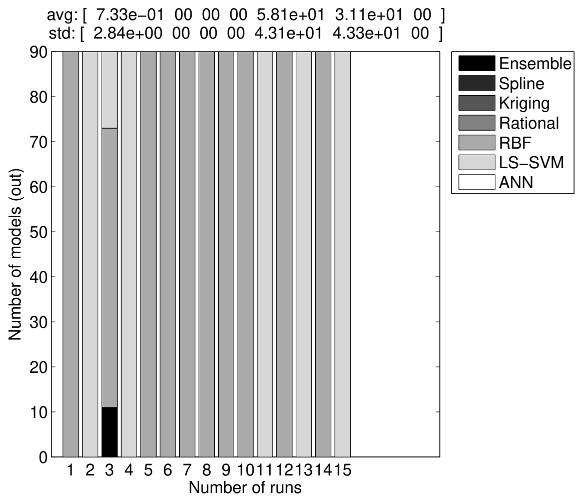
Each problem was modeled twice with the heterogeneous evolution algorithm (once with $EP = true$, once with $EP = false$) and once with homogeneous evolution (a single model type run for each model type in the HGA). To smooth out random effects each run was repeated 15 times. This resulted in a total of 516 runs which used up a total of at least 130 days worth of CPU time (excluding initial tests and failed runs). All experiments were run on CalcUA, the cluster available at the University of Antwerp, which consists of 256 Sun Fire V20z nodes (dual AMD Opteron with 4 or 8 GB RAM), running SUSE linux, and Matlab 7.6 R2008a. Due to space considerations, only the results for the *S11* (EE), and *lift* (LE) outputs are considered in this paper. For all examples the input space is normalized to the interval $[-1, 1]$.

10. Discussion

We now discuss the results of each problem separately in the following subsections.

10.1 Ackley Function

The composition of the final population for each run is shown in Figure 4 for Extinction Prevention (EP) equal to *true* and *EP=false*. The title above each sub figure shows the average and standard



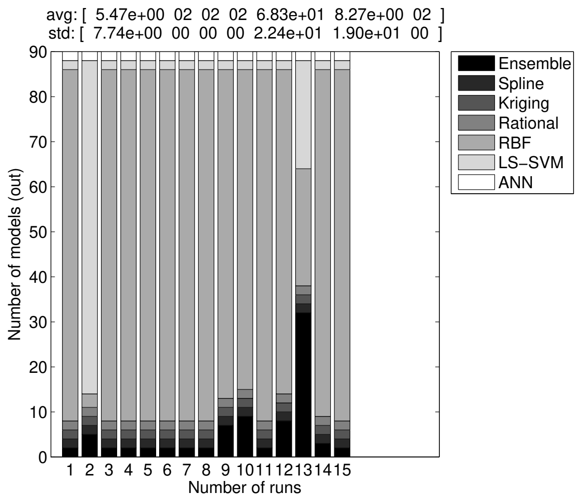
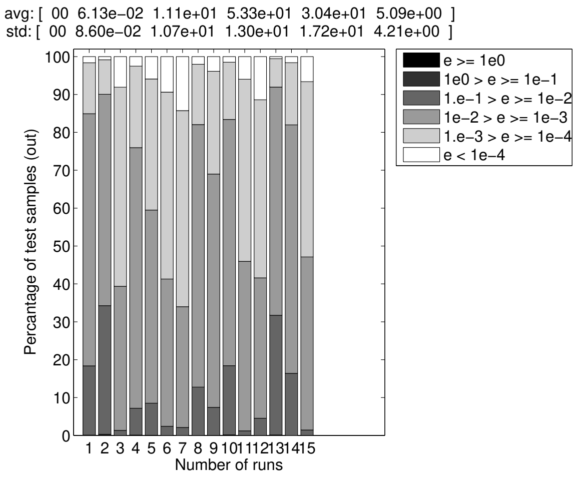


Figure 4: AF: Composition of the final population (Left: *EP=false*, Right: *EP=true*)



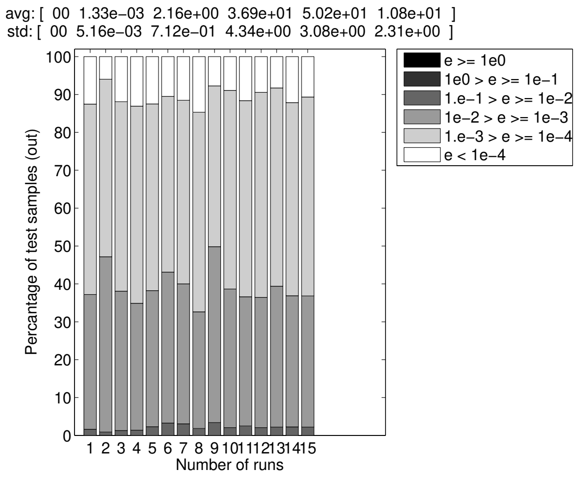


Figure 5: AF: Error histogram of the final best model in each run (Left: *EP=false*, Right: *EP=true*)

deviation over all runs. The first element of each vector corresponds to the first (top) legend entry. The error histogram of the final model of each run on the test data is shown in Figure 5. The population evolution for the run that produced the best model in both cases is shown in Figure 6. Figure 7 then depicts the evolution of the relative error (calculated according to Equation 2) on the test set as modeling progresses (again in both cases, for the run that produced the best model). The lighter the regions in Figure 7, the larger the percentage of test samples that have low relative error (RE) (according to Equation 2).

$$RE(y,\tilde{y}) = \frac{|y-\tilde{y}|}{1+|y|}. \tag{2}$$

Finally, a summary of the results for each run is shown in Table 1. The table shows the number of samples used ($|X|$), the validation error (VE), the test set error (TE), and the run time for each

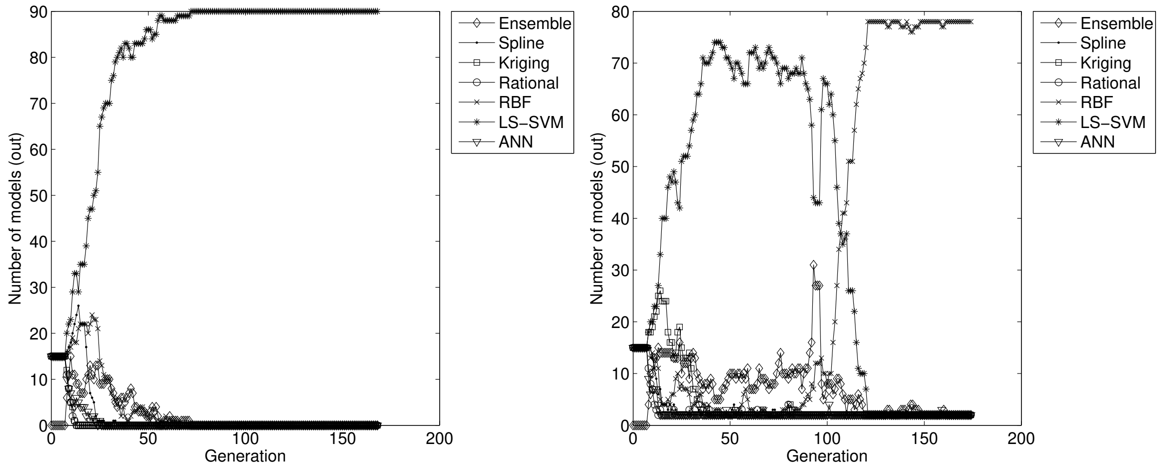


Figure 6: AF: Population evolution of the best run (Left: *EP=false*, Right: *EP=true*)

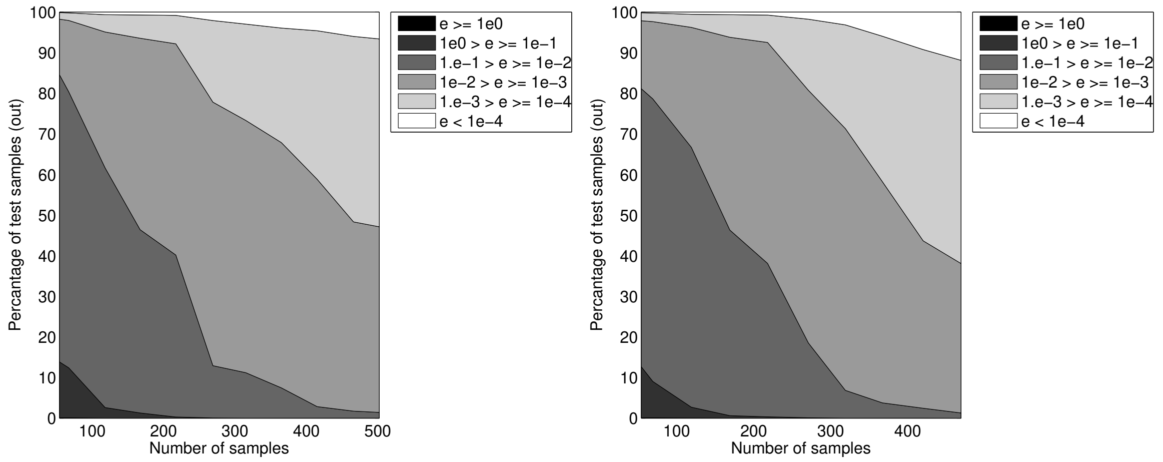


Figure 7: AF: Error evolution of the best run (Left: *EP=false*, Right: *EP=true*)

Method	$\lvert X\rvert$	σ	$\bar{VE}_{RRSE}$	σ	$\bar{TE}_{RRSE}$	σ	$\bar{time}$ *(min)*	σ
ANN	4.973E+02	1.759E+01	1.308E-02	3.428E-03	1.298E-02	3.456E-03	1.810E+02	3.176E+01
Kriging	5.263E+02	1.193E+01	2.014E-02	4.266E-03	2.038E-02	4.521E-03	7.256E+01	9.194E+00
LS-SVM	5.202E+02	1.200E+01	1.367E-02	2.262E-03	1.375E-02	2.314E-03	3.995E+01	3.276E+00
Rational	5.170E+02	1.023E+01	1.881E-01	5.854E-02	1.861E-01	5.932E-02	2.033E+01	2.221E+00
RBF	5.193E+02	1.540E+01	1.326E-02	2.365E-03	1.324E-02	2.313E-03	4.371E+01	4.133E+00
Splines	5.308E+02	1.360E+01	2.471E-02	5.660E-03	2.428E-02	5.404E-03	4.292E+01	4.984E+00
$HGA_{EP=false}$	5.055E+02	6.512E+00	3.142E-02	1.777E-02	3.094E-02	1.711E-02	2.366E+02	1.566E+02
$HGA_{EP=true}$	5.040E+02	0.000E+00	1.346E-02	1.936E-03	1.367E-02	1.897E-03	3.696E+02	6.175E+01

Table 1: AF: Comparison with homogeneous evolution

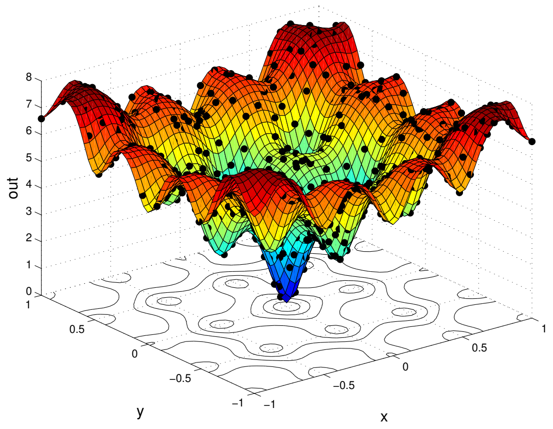


Figure 8: AF: normalized plot of the best model overall ($HGA_{EP=true}$)

experiment. All entries are averaged over 15 runs with the standard deviation shown in the adjacent column. The plot of the best model found overall is shown in Figure 8.

Regarding the composition of the final population in Figure 4, we see that the results are somewhat mixed for *EP=false.* In some runs RBF models perform best, in others LS-SVM models. This is also reflected in the corresponding error histogram plot in Figure 5. The quality of the best model found in each run differs considerably between runs. In contrast, for *EP=true*, the results are more clear cut, RBF models dominate in 14 of the 15 runs. This already demonstrates the usefulness of extinction prevention. Due to randomness in the initial population and genetic operators a model type may be driven extinct, unable to return. EP prevents this. In this particular case LS-SVM models generally perform best initially, pushing the RBF models out of the population. However, as more data becomes available (active learning), and as the hyperparameter optimization continues, superior RBF models are discovered and quickly take over the population. This is also nicely shown in Figure 6. In both cases the RBF models are driven out of the population around generation 50. Though in the *EP=true* case the RBF models are able to make a re-appearance around generation 100.

Of course nothing prevents this process from recurring. The fact that the optimal solution changes with time is not a disadvantage and should actually be expected since the optimization landscape is dynamic (due to the incremental sampling). Without EP these oscillations are impossible and everything depends on the initial conditions. As a result the danger of converging to a poor local optimum is considerably greater. Given the form of the Ackley function, we should really not be surprised that the RBF models end on top. The different radial basis functions that make up the RBF model (= a local model) can be expected to match up quite well with the ‘bumps’ of the Ackley function.

If we assess the quality of the final models (Figure 7) we see that it performs very well. After 500 samples the model has an error smaller than 0.01 on 98% of the test samples. More importantly, these results are consistent as can be seen from the *EP=true* plot in Figure 5. Actually, from an application standpoint consistency at this level (accuracy) is more important than consistency in model type selection. Since at the end of the day, from an application perspective, the accuracy of the model is typically most important, not its type.

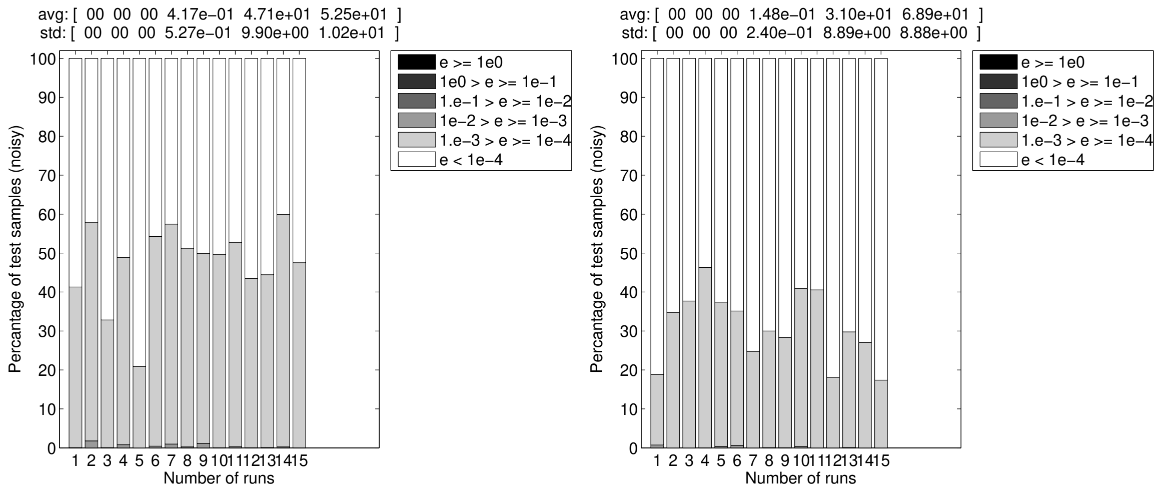


Figure 9: KF: Error histogram of the final best model in each run (Left: *EP=false*, Right: *EP=true*)

Method	$\lvert X \rvert$	σ	$\overline{VE}_{RRSE}$	σ	$\overline{TE}_{RRSE}$	σ	$\overline{time}$ *(min)*	σ
ANN	1.413E+02	2.200E+01	8.102E-04	1.614E-04	8.456E-04	1.671E-04	6.055E+01	1.258E+01
Kriging	5.200E+02	0.000E+00	1.989E-03	3.541E-04	2.023E-03	3.619E-04	1.170E+02	6.224E+01
LS-SVM	5.106E+02	2.849E+00	1.187E-01	4.967E-03	1.194E-01	5.304E-03	5.674E+01	3.571E+00
Rational	1.478E+02	9.146E+01	5.880E-04	2.115E-04	6.276E-04	2.469E-04	1.191E+01	7.547E+00
RBF	5.200E+02	0.000E+00	1.072E-01	3.620E-03	1.089E-01	4.410E-03	9.187E+01	2.157E+01
$HGA_{EP=false}$	3.059E+02	2.071E+02	1.071E-03	3.396E-04	1.085E-03	3.253E-04	3.034E+02	2.656E+02
$HGA_{EP=true}$	6.267E+01	1.486E+01	7.009E-04	2.288E-04	7.096E-04	2.024E-04	5.490E+01	3.533E+01

Table 2: KF: Comparison with homogeneous evolution

The natural question that remains, is how do these results compare with simply doing multiple homogeneous evolution (single model type, using the same GA settings) runs, one for each type separately? Those results are shown in Table 1. Studying the table we see that the HGA compares favorably. The accuracy of the final models are the essentially the same as those found by the best performing single model type run, while the variance on the results tends to be lower (EP=true). Of course this is paid for by an increase in computation time due to the increased population size of the HGA. Still, the HGA has a factor of 6 larger population size (90 vs 15) but requires only double the running time of the best performing homogeneous run (ANN). Also the total HGA running time is still less than the combined run time of all homogeneous runs.

10.2 Kotanchek Function

The composition of the final population and final error histograms for each run are shown in Figures 9 and 10. The population evolution and corresponding error evolution for the best run are shown in Figures 11 and 12. The comparison with homogeneous evolution is shown in Table 2.

The Kotanchek function is an interesting example since the GA has to ‘discover’ that 3 of the 5 variables are irrelevant. Considering the composition of the final population the Kriging functions

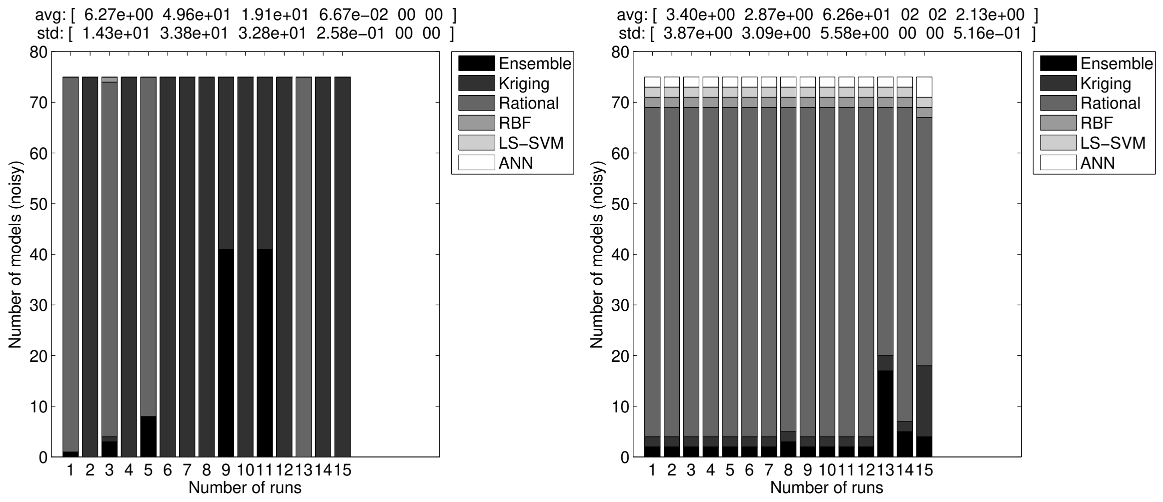


Figure 10: KF: Composition of the final population (Left: *EP=false*, Right: *EP=true*)

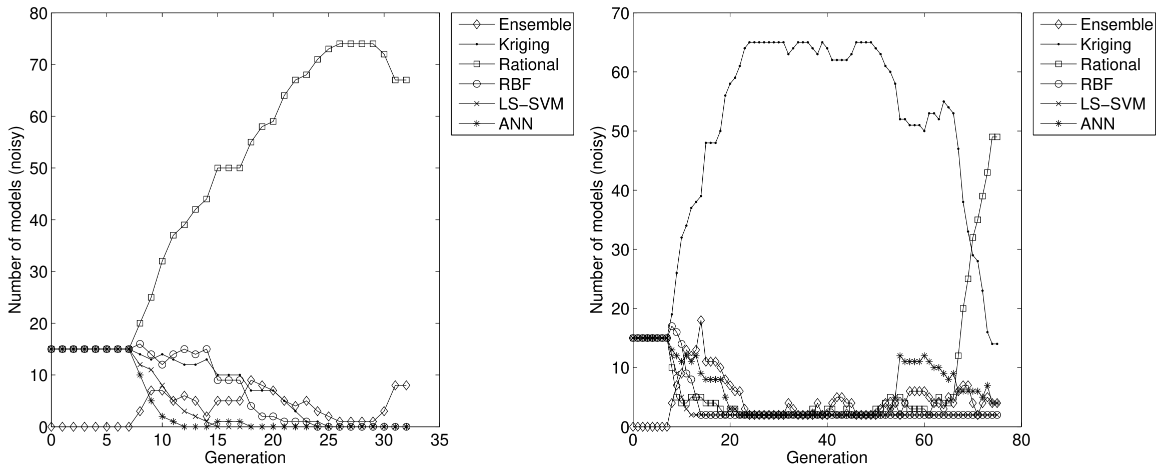


Figure 11: KF: Population evolution of the best run (Left: *EP=false*, Right: *EP=true*)

seem to be able to do this best in the *EP=false* case, with sporadic 'wins' for rational functions. In the *EP=true* case the situation is different, rational functions dominating all 15 runs. The fact that the rational functions succeed in doing this is thanks to a weighting scheme used in the genetic operators and described further in Hendrickx et al. (2006).

The usefulness of EP is demonstrated again as well. While the results of the best run for *EP=false* are better than the best run for *EP=true* (less samples), the former is much more a product of chance than the latter (which has lower variance). *EP=true* should still be preferred as it is more robust. Finally, the quality of the final models is excellent in all runs, and the performance and running time of the HGA remains competitive with the single model type runs.

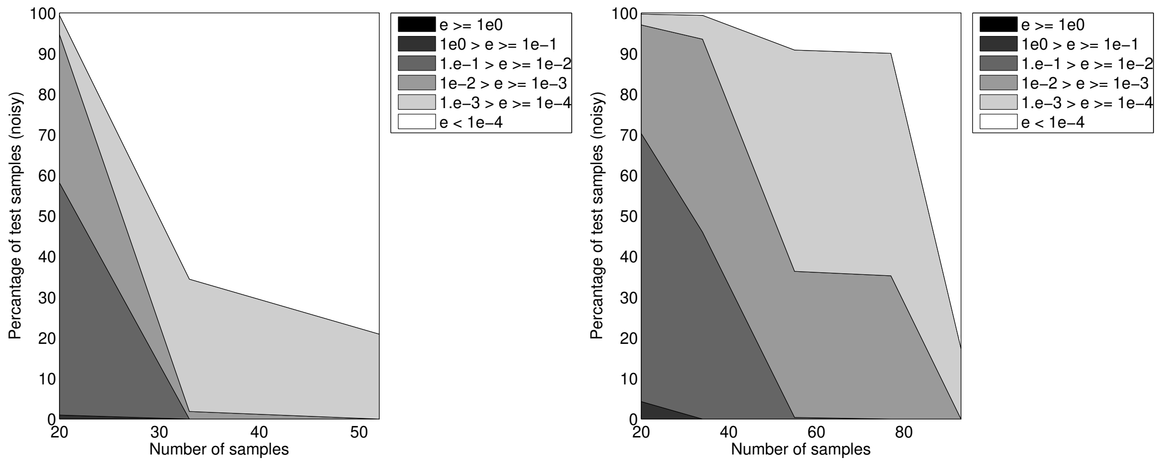


Figure 12: KF: Error evolution of the best run (Left: *EP=false*, Right: *EP=true*)

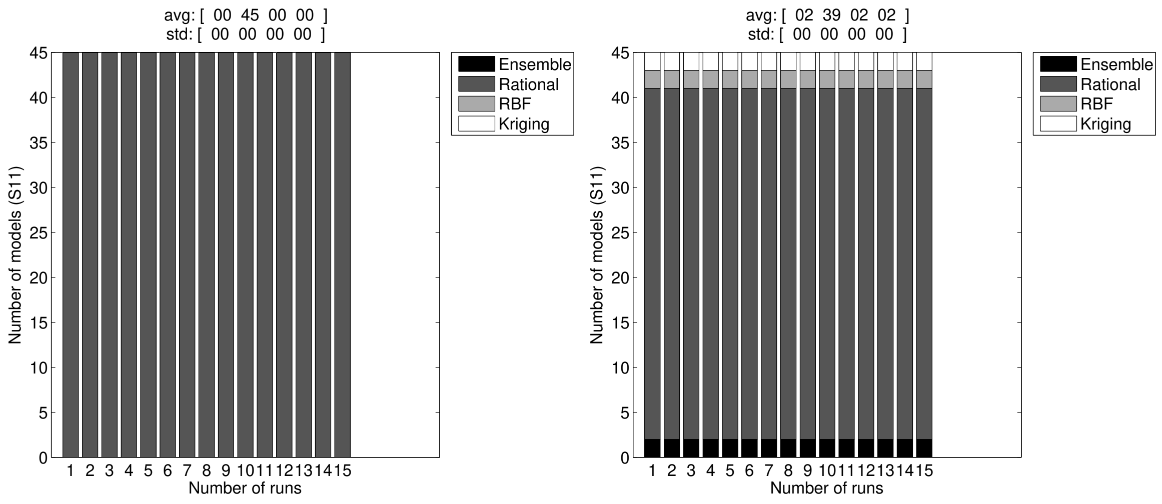
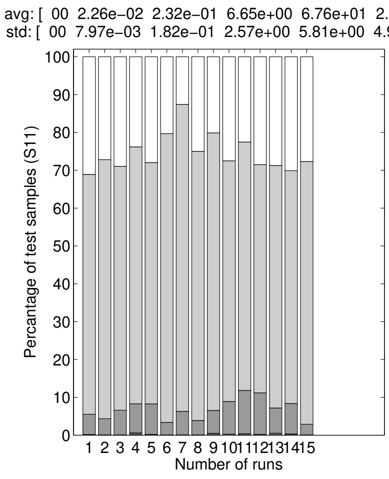


Figure 13: EE: Composition of the final population (Left: *EP=false*, Right: *EP=true*)

10.3 EM Example

The composition of the final population for each run is shown in Figure 13, and the associated error histogram in Figure 14. The population evolution and corresponding error evolution for the best run are shown in Figures 15 and 16. Table 3 summarizes the results and a plot of the best model can be found in Figure 17. Note that Table 3 shows the cross validation error (CV) instead of the validation error.

The results are very clear cut, rational functions dominate in every run, easily reaching the accuracy requirements in about 200 data points (with the *EP=true* runs generally reaching higher accuracies). This is to be expected. The physical behavior of two inductive posts in a rectangular waveguide is well described by a quotient of two differential multinomials (= the transfer function) and it is this function that needs to be modeled. Thus it is not surprising that rational functions do well since their form fits the underlying function.



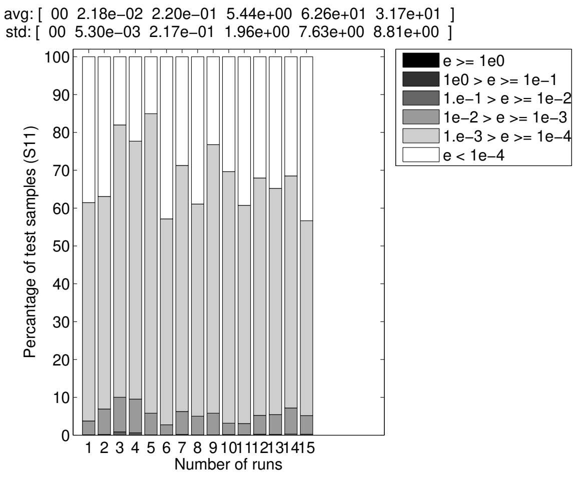
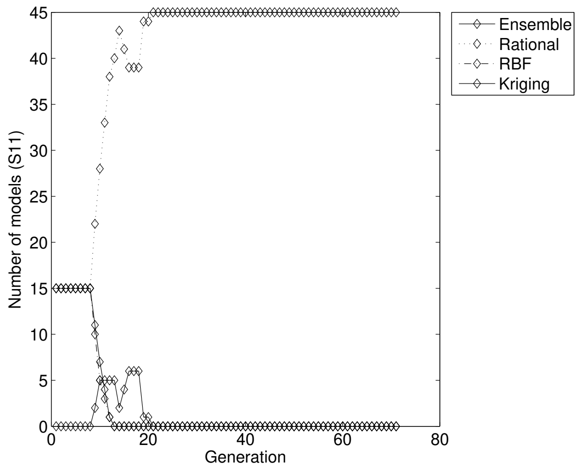


Figure 14: EE: Error histogram of the final best model in each run (Left: *EP=false*, Right: *EP=true*)



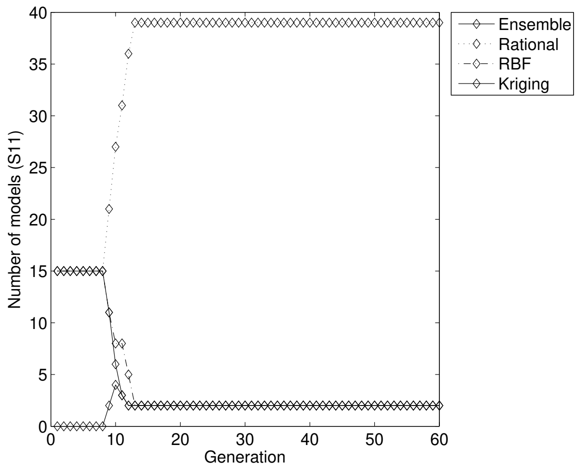


Figure 15: EE: Population evolution of the best run (Left: *EP=false*, Right: *EP=true*)

If we compare the HGA runs with the single model type runs we see significant improvements. Interestingly, the HGA runs need roughly 33-25% less sample evaluations to reach the target accuracy, and do so in a fraction of the time (less then 8 minutes vs. an average of 43 minutes for the homogeneous runs). Thus here we have a strong case for the use of the HGA.

10.4 LGBB Example

The composition of the final population for each run is shown in Figure 18 and the associated error histogram in Figure 19. The population evolution of the best run is shown in Figure 20. Table 4 shows the comparison with the homogeneous runs. A plot of the response can be found in Figure 21.

Adaptive sampling was switched off for the LGBB example. The objective was to see what accuracy can be reached and what model type prevails within a fixed time budget. The LGBB

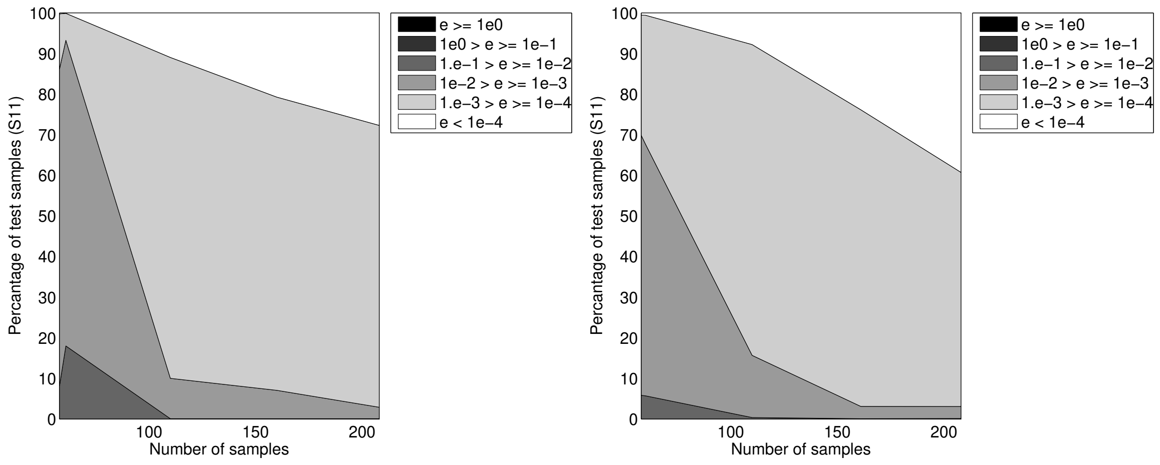


Figure 16: EE: Error evolution of the best run (Left: *EP=false*, Right: *EP=true*)

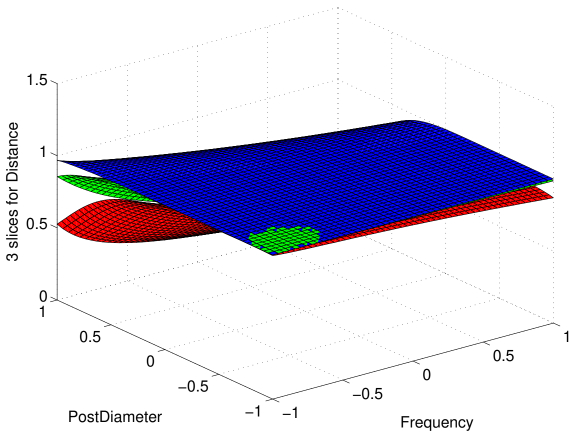
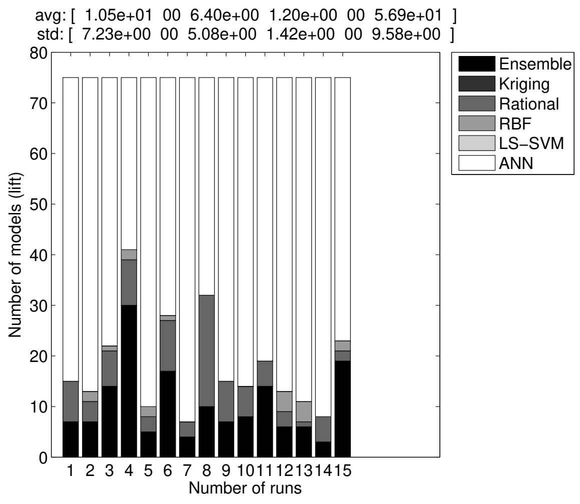


Figure 17: EE: normalized plot of $|S_{11}|$ of the best model overall ($HGA_{EP=true}$, 3 slices for $Distance$)

Method	$\|X\|$	σ	$\bar{CV}_{RRSE}$	σ	$\bar{TE}_{RRSE}$	σ	$\bar{time}$ *(min)*	σ
Kriging	7.980E+02	1.137E+02	8.541E-03	7.926E-04	1.881E-02	2.833E-03	7.202E+01	2.780E+01
Rational	8.147E+02	2.314E+02	1.152E-02	4.128E-03	1.708E-02	4.976E-03	4.046E+01	1.914E+01
RBF	6.080E+02	4.226E+01	8.123E-03	6.333E-04	1.556E-02	2.403E-03	1.713E+01	2.159E+00
$HGA_{EP=false}$	1.880E+02	2.536E+01	6.297E-03	1.770E-03	3.518E-02	4.571E-02	7.907E+00	1.440E+00
$HGA_{EP=true}$	1.980E+02	2.070E+01	6.733E-03	1.457E-03	2.227E-02	1.149E-02	7.722E+00	1.079E+00

Table 3: EE: Comparison with homogeneous evolution

example consists of a 3 dimensional data set and unlike the AF and EE examples there are no clues as to which model type is most adequate. Running the heterogeneous evolutionary algorithm it turns out that ANNs give the best fit overall (see Figure 18), achieving excellent accuracy. Changing the *EP* setting does not influence this, though the variance is lower for the *EP=true* case.



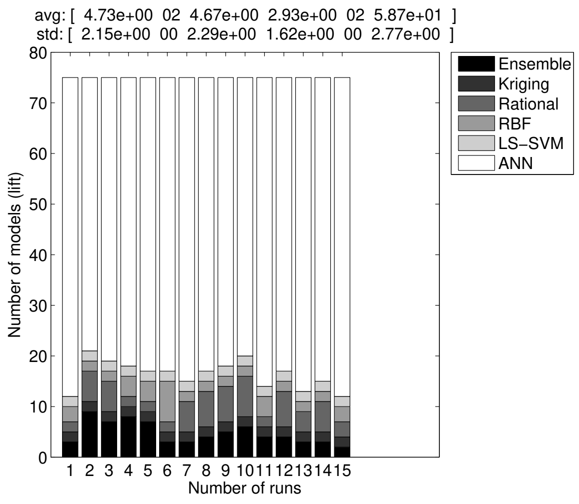
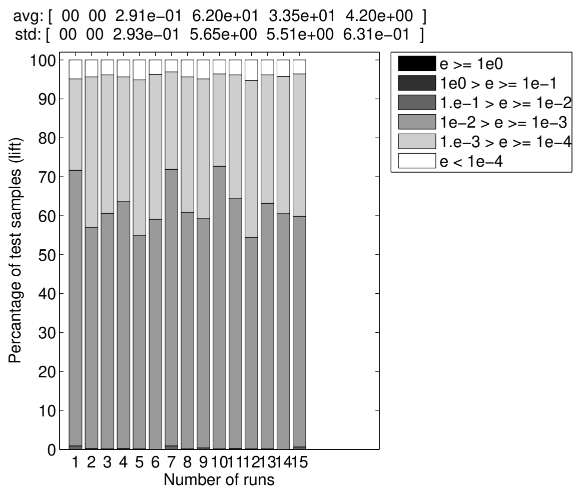


Figure 18: LE: Composition of the final population (Left: *EP=false*, Right: *EP=true*)



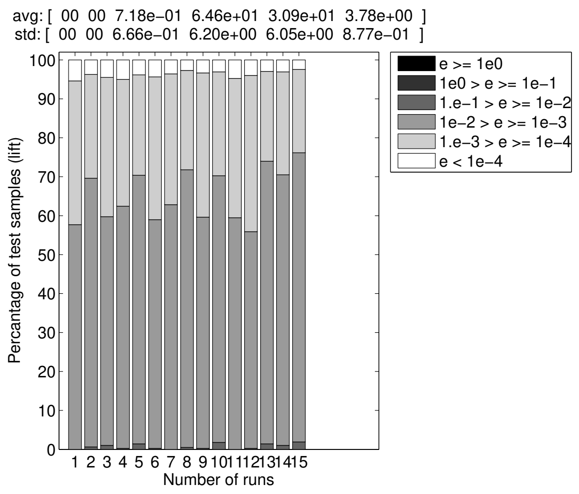


Figure 19: LE: Error histogram of the final best model in each run (Left: *EP=false*, Right: *EP=true*)

Method	$\|X\|$	$\bar{VE}_{RRSE}$	σ	*$\bar{time}$ (min)*
ANN	7.800E+02	7.47E-003	5.60E-004	3.00E+002
Kriging	7.800E+02	1.08E-001	1.96E-002	3.00E+002
LS-SVM	7.800E+02	1.40E-001	4.09E-005	3.00E+002
Rational	7.800E+02	5.20E-002	3.02E-004	3.00E+002
RBF	7.800E+02	7.34E-002	3.53E-008	3.00E+002
$HGA_{EP=false}$	7.800E+02	7.68E-003	4.38E-004	3.00E+002
$HGA_{EP=true}$	7.800E+02	7.59E-003	4.26E-004	3.00E+002

Table 4: LE: Comparison with homogeneous evolution

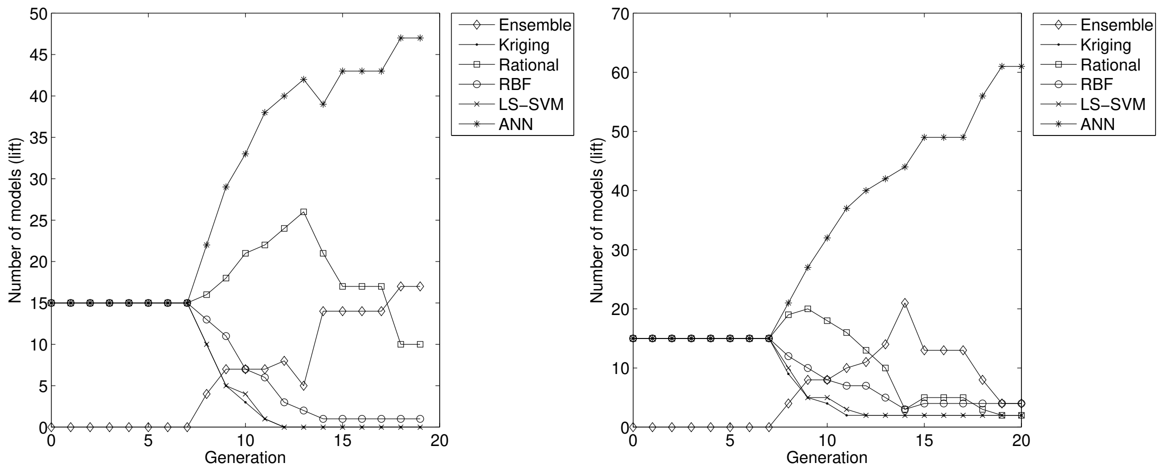


Figure 20: LE: Population evolution of the best run (Left: *EP=false*, Right: *EP=true*)

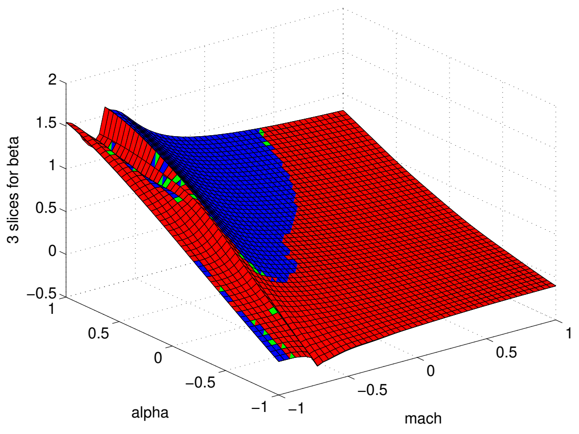


Figure 21: LE: normalized lot of the best model overall ($HGA_{EP=true}$, 3 slices for *beta*)

Within the same time limits the models produced by the HGA are comparable in accuracy to the best performing homogeneous runs, which again demonstrates the usefulness of the HGA.

Interestingly it turns out that the third dimension is negligible, the three slices in Figure 21 almost coincide. This was confirmed by using the SUMO model browser to fully explore the response. Thus we can safely conclude that side-slip angle has little or no effect on the lift on re-entry of the LGBB into the atmosphere.

10.5 Boston Housing Example

The final example is the Boston Housing data set, adaptive sampling was also switched off. The composition of the final population for each run is shown in Figure 22 and the associated error histogram in Figure 23. The population evolution of the best run is shown in Figure 24. Table 5 shows the comparison with the homogeneous runs.

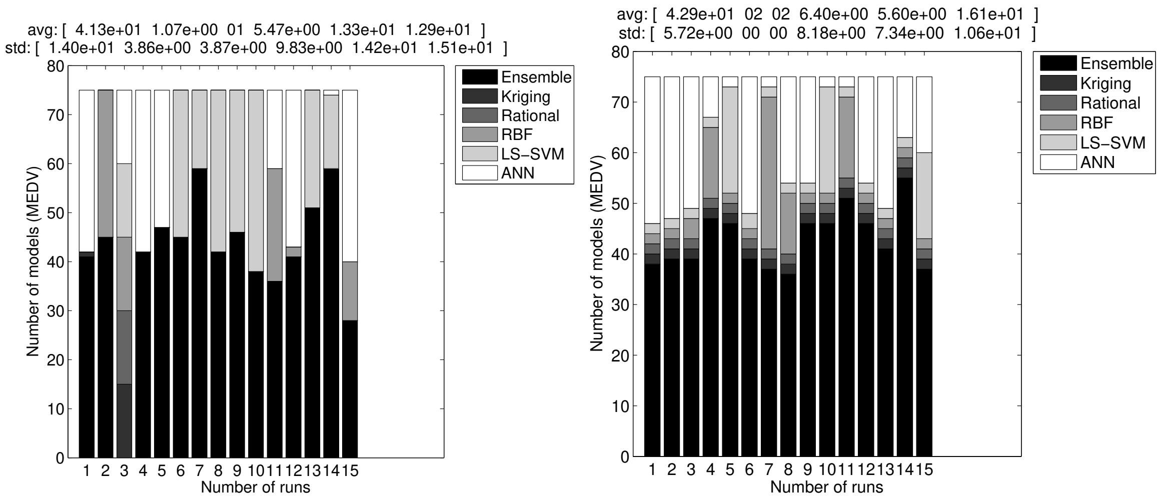


Figure 22: BH: Composition of the final population (Left: *EP=false*, Right: *EP=true*)

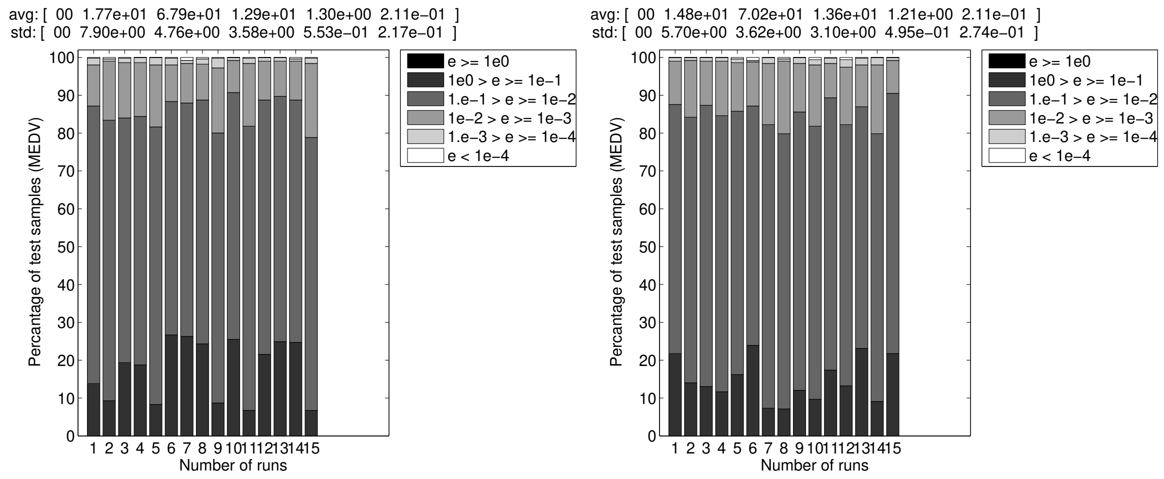


Figure 23: BH: Error histogram of the final best model in each run (Left: *EP=false*, Right: *EP=true*)

Method	$\|X\|$	$\bar{VE}_{RRSE}$	σ	*$\bar{time}$ (min)*
ANN	5.060E+02	2.985E-01	8.962E-03	1.200E+03
Kriging	5.060E+02	3.448E-01	1.077E-02	1.200E+03
LS-SVM	5.060E+02	3.421E-01	6.012E-06	1.200E+03
Rational	5.060E+02	5.006E-01	4.296E-02	1.200E+03
RBF	5.060E+02	1.228E+15	2.104E+15	1.200E+03
$HGA_{EP=false}$	5.060E+02	2.764E-01	2.768E-02	1.200E+03
$HGA_{EP=true}$	5.060E+02	2.735E-01	1.372E-02	1.200E+03

Table 5: BH: Comparison with homogeneous evolution

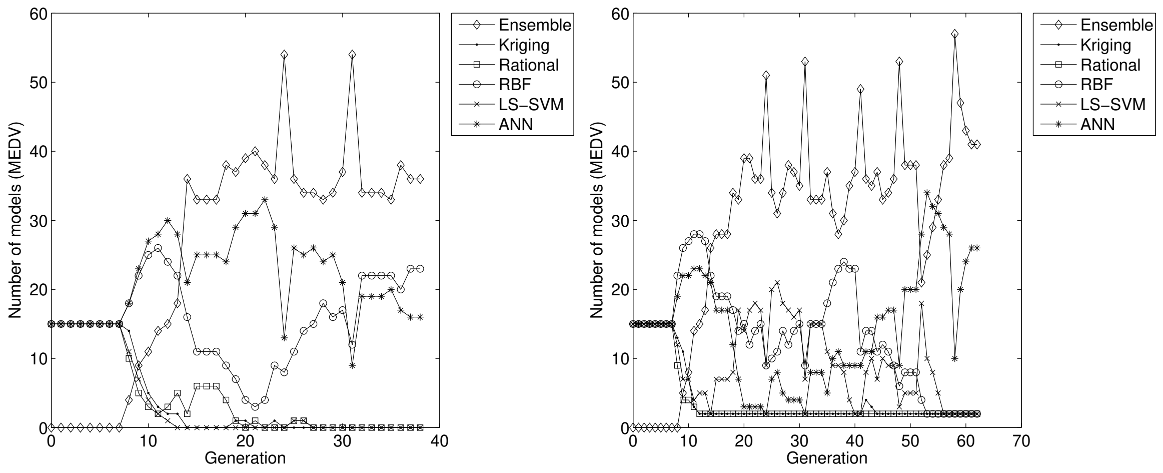


Figure 24: BH: Population evolution of the best run (Left: *EP=false*, Right: *EP=true*)

This is a somewhat curious example since it has high dimensionality (13), small support (506 tuples), and the types and ranges of the different inputs parameters vary greatly (e.g., input 4 (CHAS) is a boolean variable that is 1 if the tract borders the river and 0 otherwise while input 5 (NOX) is the nitric oxide concentration). Consequently, any analysis of this data should be preceded by a thorough statistical treatment (feature selection, variance analysis, etc.). We explicitly chose not to do this but take the data as is and treat is as black box regression problem.

The results are mixed (*see* Figure 22), though the HGA runs again outperform the homogeneous runs. (LS-)SVM and ANN models seem to be preferred over Kriging and RBF models but there is no evidence to distinguish between the models any further. Striking, though, is that about half of the final population consists of ensembles and that most of these ensembles turn out to be {ANN, RBF} pairs or multiple ANNs. Figure 25 shows the evolution of the composition of the best performing ensemble. The popularity of ensembles in this case is in line with the authors' previous experiences. When the individual model types are having trouble to fit a difficult response with none really performing much better than the other, hybrids (ensembles) tend to do well since they can produce more complicated responses. It is a signal that none of the included model types are really fit for the approximation problem.

Also striking (and interesting) are the oscillations in the population evolution (*see* Figure 24, or Figure 26 for a more marked example). It turns out that every run shows these oscillations between ensembles and one or two other model types. Interestingly these 'spikes' occur every 10 or 7 generations. It remains unclear to the authors how these oscillations may be explained. This is an issue that is being investigated in more detail.

11. Summary

In summary the results for the different test problems are very promising and in line with previous results (Gorissen, 2007; Gorissen et al., 2008a; Couckuyt et al., 2009; Gorissen et al., 2009a). The results show a consensus about which model type to use in all test cases (ignoring the BH example for the moment). In the case the consensus is not absolute (e.g., the *EP=true* run for the AF in Figure 4) the final model accuracies are essentially the same thus this is not really a problem from

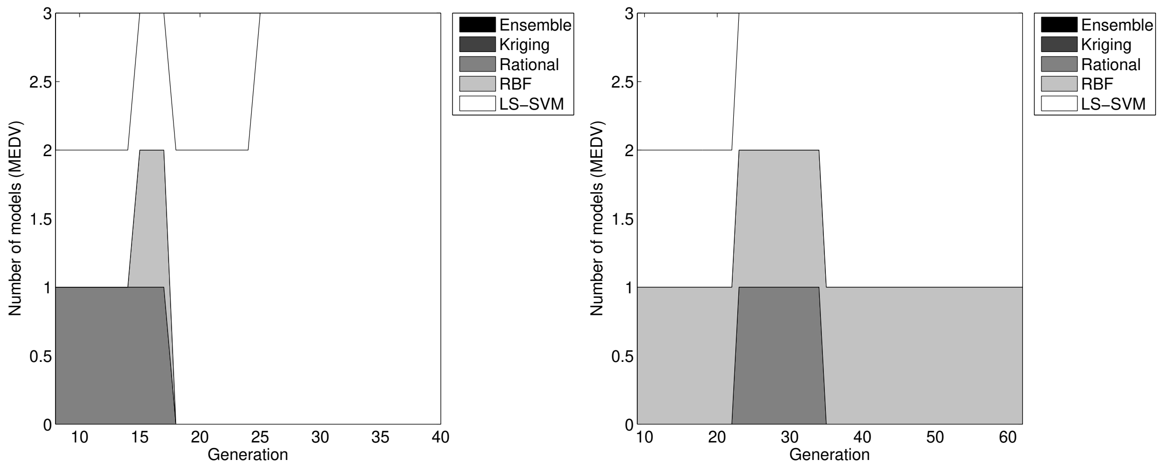


Figure 25: BH: Composition of the best performing ensemble of the best run (Left: *EP=false*, Right: *EP=true*)

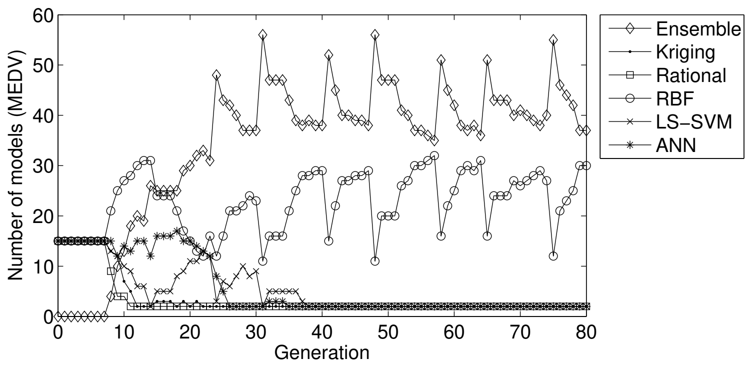


Figure 26: BH: Oscillations in the population evolution ($HGA_{EP=true}$)

an application standpoint. More important is that the target accuracy has been reached and that all model types have been given a fair chance without having to resort to a brute force approach.

In general we found the algorithm to be quite consistent across many runs. When variation does show up in the model selection results it typically is because two or more model types can fit the data equally well with only a minor difference in accuracy. This means that the GA may alternate between the different local optima, giving different model selection results, but still reaching the targets. The other reason is if the data is simply too difficult to fit using the methods included in the evolution. In this case ensembles may tend to do well. The BH example seems indicative of this situation.

It is important to remind the reader, though, that the overall performance of the HGA will of course depend on the quality of the model types themselves, and more importantly, on the quality of the creation function and genetic operators (and adequacy of the chosen representation). Good results have already been obtained with the current implementations though there is still room for improvement with respect to the application by an expert in any one type.[5] Luckily, an advantage of the HGA based approach is that since the general algorithm is now in place, it becomes possible to focus on such improvements without requiring changes to the HGA itself. Specific improvements (e.g., devised by an expert in a certain model type) are straightforward to integrate into the existing genetic operators allowing an accumulation knowledge that will improve the overall quality of the models produced by the HGA.

A next step is to further increase the number of test problems and, more importantly, investigate the influence of the different parameters involved. In this respect the migration interval and migration topology parameters are particularly important since they determine how the different model types interact. For example, if the migration interval is set too high, each deme will produce high quality models (most of the time is spent optimizing the parameters of a single model type) but there will have been very little competition between models. If it is set too low, the converse is true. More research is needed to to better understand this balance and investigate the impact of genetic drift.

12. Conclusion and Future Work

Due to the computational complexity of current simulation codes, the use of global surrogate modeling techniques (adaptive sampling, adaptive modeling) has become standard practice among scientists and engineers alike. However, a recurring problem is selecting the most adequate surrogate model type and associated complexity. In this contribution we explored an approach based on the evolutionary migration model that can help tackle this problem in an automatic way if little information is known about the true response behavior and there are no a priori model type requirements. In addition, we have illustrated the usefulness of extinction prevention and ensemble based recombination. Extinction prevention is a straightforward algorithm that prevents a species from disappearing from the gene pool at the expense of a minor cost (keeping 2 extra individuals per species ‘alive’). As a result, the optimal solution is able to change with time, making for a more flexible and adaptive system which, as demonstrated in the different examples, gives better and more consistent results.

Future work will consist of investigating the oscillations in the BH example, exploring different GA parameter values (role of the migration frequency, migration topology, etc.), incorporating more model types, and more advanced ensemble methods (e.g., stronger constraints on ensemble composition). As mentioned above, improvements to the genetic operators are ongoing in order to get more out of each model type. The utility of adding a penalty to the fitness function proportional to the model complexity and/or training time will also be investigated. Furthermore, we have been experimenting with sampling strategies that vary dynamically depending on the remaining sample budget and quality & type of surrogate currently used in the modeling process. The idea is to work towards an optimal interplay between sampling and modeling. E.g., initially the focus should be on exploration of the design space while, as accuracy of the models improves, the focus should shift towards refining the model in places where it is uncertain and ensuring the optima it exhibits are really true optima. Likewise, we are experimenting with dynamic model selection criteria. For example, if only little data is available cross validation type measures may be unreliable and it makes little sense enforcing problem specific constraints (e.g., the model response should be bounded between given bounds). However, when the data density is sufficiently high the opposite will be true. Thus there seems to be some intuition advocating the use of annealing type strategies.

5. There is a trade-off involved here. Expert application of a surrogate model type will invariably lead to problem specific bias, reducing the performance on other problems.

In addition to the global modeling use case. We are also experimenting with linking the HGA described here with the EGO framework from Jones et al. (1998). The structure of the SUMO Toolbox allows natural linking of these two components. This allows for automatic model type switching during optimization (any model type that supports prediction variance can be used) and may be beneficial for computational expensive codes.

Finally it should be noted that all the algorithms and examples described here are available for download at `http://www.sumo.intec.ugent.be`.

Acknowledgments

The authors would like to thank NASA and Robert Gramacy from the University of California, Santa Cruz for providing the data on the LGBB example. Special thanks also to Robert Lehmensiek and Petrie Meyer from the University of Stellenbosch, South Africa, for providing the simulation code for the EM example.

This work was supported by the Fund for Scientific Research Flanders (FWO Vlaanderen) and the Science and Innovation Administration Flanders (BSTC project).

References

E. Alba and M. Tomassini. Parallelism and evolutionary algorithms. *IEEE Trans. Evolutionary Computation*, 6(5):443–462, 2002.

C. A. Ankenbrandt. An extension to the theory of convergence and a proof of the time complexity of genetic algorithms. In *Foundations of Genetic Algorithms*, pages 53–68, 1990.

Y. B., R. El-Yaniv, and K. Luz. Online choice of active learning algorithms. *Journal of Machine Learning Research*, 5:255–291, 2004.

D. Busby, C. L. Farmer, and A. Iske. Hierarchical nonlinear approximation for experimental design and statistical data fitting. *SIAM Journal on Scientific Computing*, 29(1):49–69, 2007. doi: 10.1137/050639983.

P-W. Chen, J-Y. Wang, and H-M. Lee. Model selection of SVMs using GA approach. In *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, volume 3, pages 2035–2040, 25-29 July 2004.

V. Chen, K-L. Tsui, R. Barton, and M. Meckesheimer. A review on design, modeling and applications of computer experiments. *IIE Transactions*, 38:273–291, 2006.

H-S. Chung and J. J. Alonso. Comparison of approximation models with merit functions for design optimization. In *8th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Long Beach, CA, September 2000. AIAA Paper 2000-4754.

S. Conti and A. O'Hagan. Bayesian emulation of complex multi-output and dynamic computer models. research report no. 569/07, submitted to Journal of Statistical Planning and Inference. Technical report, Department of Probability and Statistics, University of Sheffield, 2007.

I. Couckuyt, D. Gorissen, H. Rouhani, E. Laermans, and T. Dhaene. Evolutionary regression modeling with active learning: An application to rainfall runoff modeling. In *International Conference on Adaptive and Natural Computing Algorithms*, volume LNCS 5495, pages 548–558, 2009.

K. Crombecq, D. Gorissen, L. De Tommasi, and T. Dhaene. A novel sequential design strategy for global surrogate modeling. In *Proceedings of the 41th Conference on Winter Simulation, accepted*, 2009.

D. Deschrijver and T. Dhaene. Rational modeling of spectral data using orthonormal vector fitting. In *Signal Propagation on Interconnects, 2005. Proceedings. 9th IEEE Workshop on*, pages 111–114, 10-13 May 2005. doi: 10.1109/SPI.2005.1500915.

M. S. Eldred and D. M. Dunlavy. Formulations for surrogate-based optimization wiht data fit, multifidelity, and reduced-order models. In *11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Protsmouth, Virginia*, 2006.

M. T. M. Emmerich, K. Giannakoglou, and B. Naujoks. Single- and multiobjective evolutionary optimization assisted by gaussian random field metamodels. *IEEE Trans. Evolutionary Computation*, 10(4):421–439, 2006.

H. J. Escalante, M. M. Gomez, and L. E. Sucar. PSMS for neural networks. In *The IJCNN 2007 Agnostic vs Prior Knowledge Challenge*, pages 678–683, 2007.

H. Jair Escalante, M. Montes, and E. Sucar. Particle swarm model selection. *Journal of Machine Learning Research (special issue on model selection)*, 10:405–440, 2008.

H. Fang, M. Rais-Rohani, Z. Liu, and M.F. Horstemeyer. A comparative study of metamodeling methods for multiobjective crashworthiness optimization. *Computers and Structures*, 83:2121–2136, 2005.

F.D. Foresee and M.T. Hagan. Gauss-newton approximation to bayesian regularization. In *Proceedings of the 1997 International Joint Conference on Neural Networks*, pages 1930–1935, 1997.

A. Forrester, A. Sobester, and A. Keane. *Engineering Design Via Surrogate Modelling: A Practical Guide*. Wiley, 2008.

F. Friedrichs and C. Igel. Evolutionary tuning of multiple svm parameters. *Neurocomputing*, 64: 107–117, 2005.

S.E. Gano, H. Kim, and D.E. Brown. Comparison of three surrogate modeling techniques: Datascape, kriging, and second order regression. In *Proceedings of the 11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, AIAA-2006-7048, Portsmouth, Virginia*, 2006.

M. Ganser, K. Grossenbacher, M. Schutz, L. Willmes, and T. Back. Simulation meta-models in the early phases of the product development process. In *Proceedings of Efficient Methods for Robust Design and Optimization (EUROMECH 07)*, 2007.

K. C. Giannakoglou, M. K. Karakasis, and I. C. Kampolis. Evolutionary algorithms with surrogate modeling for computationally expensive optimization problem. In *Proceedings of ERCOFTAC 2006 Design Optimization International Conference, Gran Canaria, Spain*, 2006.

T. Goel, R. Haftka, W. Shyy, and N. Queipo. Ensemble of surrogates. *Structural and Multidisciplinary Optimization*, 33:199–216, 2007. doi: doi:10.1007/s00158-006-0051-9.

D. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Professional, January 1989. ISBN 0201157675.

D. Gorissen. Heterogeneous evolution of surrogate models. Master's thesis, Master of AI, Katholieke Universiteit Leuven (KUL), 2007.

D. Gorissen, L. De Tommasi, J. Croon, and T. Dhaene. Automatic model type selection with heterogeneous evolution: An application to RF circuit block modeling. In *Proceedings of the IEEE Congress on Evolutionary Computation, WCCI 2008, Hong Kong*, 2008a.

D. Gorissen, L. De Tommasi, W. Hendrickx, J. Croon, and T. Dhaene. RF circuit block modeling via kriging surrogates. In *Proceedings of the 17th International Conference on Microwaves, Radar and Wireless Communications (MIKON 2008)*, 2008b.

D. Gorissen, I. Couckuyt, K. Crombecq, and T. Dhaene. Pareto-based multi-output model type selection. In *Proceedings of the 4th International Conference on Hybrid Artificial Intelligence (HAIS 2009), Salamanca, Spain*, pages 442–449. Springer - Lecture Notes in Artificial Intelligence, Vol. LNCS 5572, 2009a.

D. Gorissen, I. Couckuyt, E. Laermans, and T. Dhaene. Multiobjective global surrogate modeling,dealing with the 5-percent problem - in press. *Engineering with Computers*, 2009b.

D. Gorissen, L. De Tommasi, K. Crombecq, and T. Dhaene. Sequential modeling of a low noise amplifier with neural networks and active learning. *Neural Computing and Applications*, 18(5): 485–494, 2009c.

R. Gramacy, H. Lee, and W. Macready. Parameter space exploration with gaussian process trees. In *ICML '04: Proceedings of the 21st International Conference on Machine Learning*, page 45, New York, NY, USA, 2004. ACM Press. ISBN 1-58113-828-5. doi: http://doi.acm.org/10.1145/1015330.1015367.

L. Gu. A comparison of polynomial based regression models in vehicle safety analysis. In A. Diaz, editor, *2001 ASME Design Automation Conference, ASME, Pittsburgh, PA*, 2001.

D. Harrison and D.L. Rubinfeld. Hedonic prices and the demand for clean air. *Journal of Environmental Economics & Management*, 5:81–102, 1978.

W. Hendrickx, D. Gorissen, and T. Dhaene. Grid enabled sequential design and adaptive metamodeling. In *WSC '06: Proceedings of the 37th Conference on Winter simulation*, pages 872–881. Winter Simulation Conference, 2006. ISBN 1-4244-0501-7.

C. Hocaoglu and A. Sanderson. Planning multiple paths with evolutionary speciation. *IEEE Trans. Evolutionary Computation*, 5(3):169–191, 2001.

J. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, 1975.

R. Jin, W. Chen, and T.W. Simpson. Comparative studies of metamodelling techniques under multiple modelling criteria. *Structural and Multidisciplinary Optimization*, 23(1):1–13, December 2001. doi: 10.1007/s00158-001-0160-4.

Yaochu Jin and B. Sendhoff. Pareto-based multiobjective machine learning: An overview and case studies. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 38(3):397–415, May 2008. ISSN 1094-6977. doi: 10.1109/TSMCC.2008.919172.

Yaochu Jin, Markus Olhofer, and Bernhard Sendhoff. A framework for evolutionary optimization with approximate fitness functions. *IEEE Trans. Evolutionary Computation*, 6(5):481–494, 2002.

D. R. Jones, M. Schonlau, and W. J. Welch. Efficient global optimization of expensive black-box functions. *J. of Global Optimization*, 13(4):455–492, 1998. ISSN 0925-5001. doi: http://dx.doi.org/10.1023/A:1008306431147.

A. C. Keys, L. P. Rees, and A. G. Greenwood. Performance measures for selection of metamodels to be used in simulation optimization. *Decision Sciences*, 33:31 – 58, 2007.

J. Kleijnen, S. Sanchez, T. Lucas, and T. Cioppa. State-of-the-art review: A user's guide to the brave new world of designing simulation experiments. *INFORMS Journal on Computing*, 17(3): 263–289, 2005.

J. Knowles and H. Nakayama. Meta-modeling in multiobjective optimization. In *Multiobjective Optimization: Interactive and Evolutionary Approaches*, pages 245–284. Springer-Verlag, Berlin, Heidelberg, 2008. ISBN 978-3-540-88907-6. doi: http://dx.doi.org/10.1007/978-3-540-88908-3_10.

R. Lehmensiek. *Efficient Adaptive Sampling Applied to Multivariate, Multiple Output Rational Interpolation Models, with Applications in Electromagnetics-based Device Modeling*. PhD thesis, University of Stellenbosch, 2001.

S. Lessmann, R. Stahlbock, and S.F. Crone. Genetic algorithms for support vector machine model selection. In *Proceedings of the International Joint Conference on Neural Networks, 2006. IJCNN '06.*, pages 3063–3069, 16-21 July 2006.

X. Rong Li and Z. Zhao. Evaluation of estimation algorithms part I: incomprehensive measures of performance. *IEEE Transactions on Aerospace and Electronic Systems*, 42(4):1340–1358, October 2006. ISSN 0018-9251. doi: 10.1109/TAES.2006.314576.

D. Lim, Y-S. Ong, Y. Jin, and B. Sendhoff. A study on metamodeling techniques, ensembles, and multi-surrogates in evolutionary computation. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation (GECCO 07)*, pages 1288–1295, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-697-4. doi: http://doi.acm.org/10.1145/1276958.1277203.

S. N. Lophaven, H. B. Nielsen, and J. Søndergaard. Aspects of the matlab toolbox DACE. Technical report, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby, 2002.

D. MacKay. Bayesian model comparison and backprop nets. In J. E. Moody, S. J. Hanson, and R. P. Lippmann, editors, *Advances in Neural Information Processing Systems 4*, pages 839–846, San Mateo, California, 1992. Morgan Kaufmann.

T. Nakama. Theoretical analysis of genetic algorithms in noisy environments based on a markov model. In *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation (GECCO 08)*, pages 1001–1008, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-130-9. doi: http://doi.acm.org/10.1145/1389095.1389283.

A. Neubauer. A theoretical analysis of the non-uniform mutation operator for the modified genetic algorithm. In *Proceedings of the IEEE International Conference on Evolutionary Computation*, pages 93–96, Apr 1997. doi: 10.1109/ICEC.1997.592275.

M. Nowostawski and R. Poli. Parallel genetic algorithm taxonomy. In *Knowledge-Based Intelligent Information Engineering Systems, 1999. Third International Conference*, pages 88–92, 31 Aug.-1 Sept. 1999. doi: 10.1109/KES.1999.820127.

Y-S. Ong, P.B. Nair, and K.Y. Lum. Max-min surrogate-assisted evolutionary algorithm for robust design. *Evolutionary Computation, IEEE Transactions on*, 10(4):392–404, Aug. 2006. doi: 10.1109/TEVC.2005.859464.

I. Paenke, J. Branke, and Y. Jin. Efficient search for robust solutions by means of evolutionary algorithms and fitness approximation. *IEEE Trans. Evolutionary Computation*, 10(4):405–420, 2006.

B. Pamadi, P. Covell, P. Tartabini, and K. Murphy. Aerodynamic characteristics and glide-back performance of langley glide-back booster. In *Proceedings of 22nd Applied Aerodynamics Conference and Exhibit, Providence, Rhode Island*, 2004.

H. Pei and E. Goodman. A comparison of cohort genetic algorithms with canonical serial and island-model distributed ga's. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 501–510, San Francisco, California, USA, 7-11 July 2001. Morgan Kaufmann. ISBN 1-55860-774-9.

R. Poli. Exact schema theory for genetic programming and variable-length genetic algorithms with one-point crossover. *Genetic Programming and Evolvable Machines*, 2(2):123–163, 2001. ISSN 1389-2576. doi: http://dx.doi.org/10.1023/A:1011552313821.

X. Qi and F. Palmieri. Theoretical analysis of evolutionary algorithms with an infinite population size in continuous space. part I: Basic properties of selection and mutation. *Neural Networks, IEEE Transactions on*, 5(1):102–119, Jan 1994a. ISSN 1045-9227. doi: 10.1109/72.265965.

X. Qi and F. Palmieri. Theoretical analysis of evolutionary algorithms with an infinite population size in continuous space. part II: Analysis of the diversification role of crossover. *Neural Networks, IEEE Transactions on*, 5(1):120–129, Jan 1994b. ISSN 1045-9227. doi: 10.1109/72.265966.

N. Queipo, R. Haftka, W. Shyy, T. Goel, R. Vaidyanathan, and P. Tucker. Surrogate-based analysis and optimization. *Progress in Aerospace Sciences*, 41:1–28, 2005.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

G. Rawlins, editor. *Foundations of Genetic Algorithms*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1991. ISBN 1558605592.

R.G. Regis and C.A. Shoemaker. Local function approximation in evolutionary algorithms for the optimization of costly functions. *IEEE Trans. Evolutionary Computation*, 8(5):490–505, 2004.

S. Rogers, M. Aftosmis, S. Pandya, and N. Chaderjian. Automated CFD parameter studies on distributed parallel computers. In *Proc of.16th AIAA Computational Fluid Dynamics Conference, Orlando, Florida*, 2003.

E. Sanchez, S. Pintos, and N.V. Queipo. Toward an optimal ensemble of kernel-based approximations with engineering applications. In *In Proceedings of the International Joint Conference on Neural Networks, 2006. IJCNN '06.*, pages 2152–2158, 2006. doi: 10.1109/IJCNN.2006.246987.

T. Santner, B. Williams, and W. Notz. *The design and analysis of computer experiments*. Springer series in statistics. Springer, 2003.

A. Sharkey. On combining artificial neural nets. *Connectionist Science*, 8(3):299–314, 1996.

T. Simpson, J. D. Poplinski, P. N. Koch, and J. K. Allen. Metamodels for computer-based engineering design: Survey and recommendations. *Eng. Comput. (Lond.)*, 17(2):129–150, 2001.

T. W. Simpson, V. Toropov, V. Balabanov, and F. A. C. Viana. Design and analysis of computer experiments in multidisciplinary design optimization: a review of how far we have come or not. In *Proceedings of the 12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, 2008 MAO, Victoria, Canada*, 2008.

G. Smits and M. Kotanchek. Pareto-front exploitation in symbolic regression. In *Genetic Programming Theory and Practice II*. Springer, Ann Arbor, USA, 2004.

D. P. Solomatine and A. Ostfeld. Data-driven modelling : some past experiences and new approaches. *Journal of hydroinformatics*, 10(1):3–22, 2008.

K. Stanley and R. Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2):99–127, 2002. ISSN 1063-6560. doi: http://dx.doi.org/10.1162/106365602320169811.

M. Streeter and L. Becker. Automated discovery of numerical approximation formulae via genetic programming. *Genetic Programming and Evolvable Machines*, 4(3):255–286, 2003. ISSN 1389-2576. doi: http://dx.doi.org/10.1023/A:1025176407779.

J.A.K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle. *Least Squares Support Vector Machines*. World Scientific Publishing Co., Pte, Ltd., Singapore, 2002. ISBN 981-238-151-1.

D. J.J. Toal, N. W. Bressloff, and A. J. Keane. Kriging hyperparameter tuning strategies. *AIAA Journal*, 46(5):1240–1252, 2008.

S. Tomioka, S. Nisiyama, and T. Enoto. Nonlinear least square regression by adaptive domain method with multiple genetic algorithms. *IEEE Transactions on Evolutionary Computation*, 11 (1):1–16, February 2007.

P. Triverio, S. Grivet-Talocia, M.S. Nakhla, F. G. Canavero, and R. Achar. Stability, causality, and passivity in electrical interconnect models. *IEEE Transactions on Advanced Packaging*, 30(4): 795–808, 2007.

C. Turner, R. Crawford, and M. Campbell. Multidimensional sequential sampling for nurbs-based metamodel development. *Engineering with Computers*, 23(3):155–174, 2007. ISSN 0177-0667. doi: http://dx.doi.org/10.1007/s00366-006-0051-9.

E.J. Vladislavleva, G.F. Smits, and D. den Hertog. Order of nonlinearity as a complexity measure for models generated by symbolic regression via pareto genetic programming. *Evolutionary Computation, IEEE Transactions on*, 13(2):333–349, April 2009. ISSN 1089-778X. doi: 10. 1109/TEVC.2008.926486.

I. Voutchkov and A.J. Keane. Multiobjective Optimization using Surrogates. In I.C. Parmee, editor, *Adaptive Computing in Design and Manufacture 2006. Proceedings of the Seventh International Conference*, pages 167–175, Bristol, UK, April 2006.

G. Wang and S. Shan. Review of metamodeling techniques in support of engineering design optimization. *Journal of Mechanical Design*, 129(4):370–380, 2007. doi: 10.1115/1.2429697.

D. Whitley, S. Rana, and R. B. Heckendorn. The island model genetic algorithm: On separability, population size and convergence. *Journal of Computing and Information Technology*, 7(1):33–47, 1999.

Y. Xiong, W . Chen, D. Apley, and X. Ding. A nonstationary covariance-based kriging method for metamodeling in engineering design. *International Journal for Numerical Methods in Engineering*, 71(6):733–756, 2007.

R.J. Yang, N. Wang, C. H. Tho, and J. P. Bobinaeu. Metamodeling development for vehicle frontal impact simulation. *Journal of Mechanical Design*, 127(5):1014–1020, September 2005.

X. Yao. Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9):1423–1447, Sept. 1999. doi: 10.1109/5.784219.

X. Yao and Y. Xu. Recent advances in evolutionary computation. *Journal of Computer Science and Technology*, 21(1):1–18, 2006.

K. Ye, W. Li, and A. Sudjianto. Algorithmic construction of optimal symmetric latin hypercube designs. *Journal of Statistical Planning and Inference*, 90:145–159, 2000.

Y-S. Yeun, W-S. Ruy, Y-S. Yang, and N-J. Kim. Implementing linear models in genetic programming. *IEEE Trans. Evolutionary Computation*, 8(6):542–566, 2004.

C. Zhang, H. Shao, and Y. Li. Particle swarm optimisation for evolving artificial neural network. In *Systems, Man, and Cybernetics, 2000 IEEE International Conference on*, volume 4, pages 2487–2490vol.4, 8-11 Oct. 2000. doi: 10.1109/ICSMC.2000.884366.