Automatic Model Type Selection with Heterogeneous Evolution: An application to RF circuit block modeling

Dirk Gorissen, Luciano De Tommasi, Jeroen Croon, Tom Dhaene

***Abstract*— Many complex, real world phenomena are difficult to study directly using controlled experiments. Instead, the use of computer simulations has become commonplace as a cost effective alternative. However, regardless of Moore's law, performing high fidelity simulations still requires a great investment of time and money. Surrogate modeling (metamodeling) has become indispensable as an alternative solution for relieving this burden. Many surrogate model types exist (Support Vector Machines, Kriging, RBF models, Neural Networks, ...) but no type is optimal in all circumstances. Nor is there any hard theory available that can help make this choice. The same is true for setting the surrogate model parameters (Bias - Variance trade-off). Traditionally, the solution to both problems has been a pragmatic one, guided by intuition, prior experience or simply available software packages. In this paper we present a more founded approach to these problems. We describe an adaptive surrogate modeling environment, driven by speciated evolution, to automatically determine the optimal model type and complexity. Its utility and performance is presented on a case study from electronics.**

I. Introduction

For the majority of 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. This is especially evident for routine tasks such as optimization, sensitivity analysis and design space exploration [1].

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 model 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 metamodels or response surface models (RSM)), as opposed to the white box approach known as Model Order Reduction (MOR).

Dirk Gorissen and Tom Dhaene are with the Department of Information Technology, Ghent University, Belgium (email: {dirk.gorissen, tom.dhaene}@intec.ugent.be). Luciano De Tommasi is with the Department of Maths and Computer Science, Antwerp University, Belgium (email: luciano.detommasi@ua.ac.be) and Jeroen Croon is with the Device Modeling Department at NXP-TSMC Research Center Eindhoven, The Netherlands (email: jeroen.croon@nxp.com)

This work was supported by the Fund for Scientific Research Flanders (FWO Vlaanderen), the Science and Innovation Administration Flanders (BSTC project), and the O-MOORE-NICE! project as supported by the European Commission through the Marie Curie program under contract number MTKI-CT-2006-042477.

Note that for this paper we are concerned with *global* surrogate models as opposed to local ones (though the proposed method can be applied to local surrogates as well). Local surrogates are primarily used in optimization, where small, simple models are used to guide the search towards a global optimum. Once the optimum is reached the surrogate is discarded. In contrast, global surrogates try to accurately capture the global behavior of the system. This makes them very useful for tasks like design space exploration, sensitivity analysis, high dimensional visualization, and prototyping. Additionally, they can be embedded as a subsystem in a more complex design and optimization can still occur as a post-processing step (supporting multiple boundary conditions).

Surrogate modeling will be introduced in the next section. Section III will then discuss speciated evolution, followed by its application to surrogate modeling in section IV. Section V then describes a relevant problem from electronics which is used to evaluate the proposed approach, followed by the actual results in section VII and their discussion in section VIII. We conclude in section IX with pointers to future work.

This paper concentrates on the usefulness of heterogeneous evolution as compared to more straightforward surrogate modeling methods on a given real world example. For more in-depth background on the theory and the motivations behind our approach (including results on some benchmark problems), the reader is directed to [2].

II. Surrogate Modeling

A. Introduction

A global surrogate model is a relatively small, simple model intended to mimic the behavior of a large complex model, that is, to reproduce the 'true' model's input-output relationships [3]. Surrogate models have found universal use throughout scientific research, from geology [4] to aerodynamics [5].

The principal reason driving surrogate model use is that the simulator is too time consuming to run for a large number of simulations. One model evaluation may take many minutes, hours, days or even weeks [6]. A simpler approximation of the simulator is needed to make optimization, sensitivity analysis, design space exploration, etc. feasible. A second reason is when simulating large scale systems [7]. A classic example is the full-wave simulation of an electronic circuit. Electro-magnetic (EM) modeling of the whole circuit in one run is almost intractable. Instead the circuit is modeled as a collection of small, compact, accurate approximations that represent the different functional components (capacitors, filters, ...).

978-1-4244-1823-7/08/$25.00©2008 IEEE

B. Problem domain

The mathematical formulation of the problem is as follows: approximate an unknown multivariate function $f : \Omega \mapsto \mathbb{C}^q$, defined on some domain $\Omega \subset \mathbb{R}^d$ and whose function values $f|_X = (f(x_1),...,f(x_n)) \in \mathbb{C}^q$ are known at a fixed set of pairwise distinct sample points $X = \{x_1,...,x_n\} \subset \Omega$. Constructing an approximation then requires finding a suitable function y from an approximation space Y such that $y : \mathbb{R}^d \mapsto \mathbb{C}^q \in Y$ and y closely resembles f as measured by the approximation error $||f - y||_\nu$ according to some norm $||.||_\nu$. The task is then to find the best approximation $\tilde{y} \in Y$ such that $\tilde{y}$ satisfies $\min_{s \in S} ||f - y||_\nu = ||f - \tilde{y}||_\nu$. An additional assumption is that f is expensive to compute (thus the number of function evaluations $f(x_i)$ needs to be minimized).

While the mathematical formulation is clear cut, its practical implementation raises an obvious question: Which approximation method should be used for the function y? Many possibilities exist: rational functions, RBF models, Support Vector Machines (SVM), Kriging, multivariate splines, ... Unfortunately, little or no theory exists to help make this choice. Thus, it is usually a pragmatic one. Additionally, once an approximation method is chosen, a second problem is optimizing the model parameters (i.e., finding the optimal bias-variance trade-off). For example, in the case of RBF SVMs, finding the optimal C, ε and σ parameters.

We propose to tackle both problems through the use of a single Parallel Genetic Algorithm (PGA) with speciation. The idea is to maintain a heterogeneous population of surrogate model types and let evolution select the most adequate approximation technique for a given problem and (in this case, dynamically changing) data distribution. The authors believe to be the first provide an automatic means to simultaneously optimize the bias-variance trade-off and automatically select the most adequate model type for a given problem. The details will be presented in section IV. While the focus of this paper is regression, the approach can easily be generalized to classification problems as well.

III. Speciated Evolution

A. Introduction

From a biological standpoint it makes sense to consider *speciation* within GA: genomes that differ considerably from the rest of the population are automatically split off into a separate sub-population and continue to evolve independently. Thus forming a new species. In this way the parameter space is searched more efficiently. The terms Parallel Genetic Algorithms (PGA) or Distributed Genetic Algorithms (DGA) [8] 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 [9], [10]. In this paper we are only interested in PGA on the model level (i.e., different speciation models), parallelism for computational reasons, including scheduling considerations, will not be considered (see [11] for a good reference in this respect). We very briefly discuss the main speciation models in the following subsection (see [10] for a more formal treatment).

B. Speciation models

The most popular ways of introducing speciation are the *island model* [12], the *cellular model* [13] (also known as the *diffusion* model or *massively parallel GA*) and speciation through fitness sharing [14]. Besides these, many variations exist such as: overlapping demes, dynamic demes, segregative GA, crowding, hierarchical GA, Cohort GA, ... Additionally, many hybrid schemes are possible where different aspects of each model are combined. The different speciation models have found widespread use throughout many domains. Examples can be found in machine learning [15], surrogate driven optimization [16] (using a hierarchy of island models), and vehicle concept selection in aerospace [17]. Excellent references can be found in [18].

In this paper we are primarily concerned with the *island model*, probably the most well known PGA. Different sub-populations exist and sporadic migration can occur between islands allowing for the exchange of genetic material between species and inter-species competition for resources. This model is also known as the *migration model* or *stepping stone model*, depending on the migration constraints. As stated above, the population is divided into a number of independent *demes*, with inter-deme migration. Selection and recombination are restricted per deme, such that each sub-population may evolve towards different locally optimal regions of the search space (*niches*). Depending on the size and number of demes the model can be *coarse grained* or *fine grained* [9]. 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.

IV. Heterogeneous Surrogate Evolution

A. Motivation

We propose to apply speciated evolution, in the form of the island model, to the well known statistical problem of model selection. Different model types are preferred in different domains. For example rational functions are widely used by the Electro-Magnetic (EM) community [19] while ANN are preferred for hydrological modeling [20]. Differences in model type usage are mainly due to practical reasons (lack of expertise, time, available software, ...) [1]. Also it should be noted that 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 expert who built it. 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. There is ample literature available

[1]In some cases, however, 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 (LTI) systems since theory is available that can be used to prove that rational pole-residue models conserve certain physical quantities (see [21]).

that benchmarks different model types on toy problems [22], [23], [24], [25]. But claims that a particular model type is superior to others should always be met with some skepticism (cfr. *No-Free-Lunch-Theorems*).

Selecting the model type is only part of the problem, the complexity of the model needs to be chosen as well. Each model type has a set of tunable parameters θ that control the complexity of the model M and thus the bias-variance tradeoff. In general, finding the optimal bias-variance trade-off is a difficult optimization problem with many local minima. Again, no method can be said to be the best overall.

Thus, 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. In this contribution 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.

B. *Related Work*

The evolutionary generation of regression models for given input-output data has been widely studied in the Genetic Programming community [26], [27], [28]. Given a set of mathematical primitives (+, *sin, exp, /, x, y, ...*) 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: [29], [30] and the extensive work by X. Yao [31]. The same is true for the use of surrogate models in evolutionary optimization of expensive simulators (to approximate the fitness function). References include [32], [33] and the work by Ong et al. [34].

However, the authors were unable to find any evidence of the use of evolutionary algorithms for the evolution of multiple surrogate model types simultaneously (heterogeneous evolution). In all the related work that the authors considered, speciation was always constrained to one particular model type (e.g., neural networks [35]). The model type selection problem was still left as an a-priori choice for the user. The authors assume to be the first to tackle the model parameter optimization and model type selection problem in one speciated GA.

C. *Software platform*

The approach presented in this paper is based on an existing platform for adaptive surrogate modeling: the **SU**rrogate **MO**deling Toolbox (SUMO Toolbox) [36]. We briefly discuss the toolbox below.

1) Background: The SUMO Toolbox is an adaptive tool that integrates different modeling approaches and implements a fully automated, adaptive global surrogate model construction algorithm. Given a simulation engine the toolbox produces a surrogate model within the time and accuracy constraints set by the user. However, since there is no such thing as a 'one-size-fits-all', the toolbox was designed to be modular and extensible but not be too cumbersome to use or configure. Different plugins are supported: model types (rational functions, Kriging, ...), model parameter optimization algorithms (BFGS, GA, ...), sample selection (random, error based, ...), and sample evaluation methods (local, on a cluster or grid). The behavior of each component is configurable through a central XML configuration file and components can easily be added, removed or replaced by custom implementations (see figure 1).

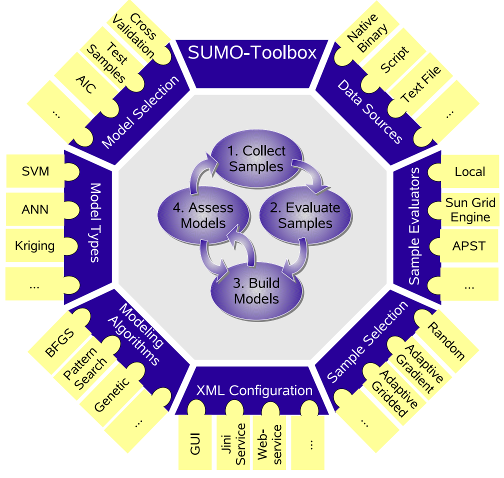

Fig. 1. SUMO Toolbox Plugins

2) Control flow: The toolbox control flow is as follows: an initial set of samples (data points) is chosen according to some experimental design. Based on this initial set, one or more surrogate models are constructed and their parameters optimized according to an optimization algorithm (e.g., pattern search or GA). Models are assigned a score based on one or more measures (e.g., cross-validation or AIC) and the hyperparameter 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 top k models (incremental sampling). The model parameter optimization resumes (or is restarted intelligently) and the whole process repeats until a threshold has been exceeded or the user required accuracy has been reached.

It is worth stressing that data points must be selected incrementally. Data is expensive to obtain (many minutes, hours or even days for one simulation) so traditional, up-front, full factorial designs are too expensive. In these settings it is more efficient to select data points incrementally, as modeling progresses, there where the potential information gain is the highest. This process is called adaptive sampling, but is also known as active learning [37], Optimal Experimental Design [38] and sequential design [39]. Consequently, adaptive sampling implies that the model parameter optimization landscape is dynamic and not static as is often assumed.

D. *Implemented PGA*

The speciation model used is the island model since it is the most natural for this problem. The algorithm is based on the Matlab GADS toolbox and works as follows (all details can be found in [2]): An initial sub-population $deme_i$ is created for each model type M_i $(i = 1,..,n)$. Then each deme is allowed to evolve according to an elitist GA. The models M_i are implemented as Matlab objects (with full polymorphism) and each model type can choose its own representation and genetic operator implementations. The fitness function calculates the quality of the model fit on a set of d-dimensional data points X, according to an error measure E. I.e., $fitness : \mathbb{R}^d \times \mathbb{C}^q \mapsto \mathbb{R}$. Parents are selected according to some selection algorithm (e.g., tournament selection) and offspring are generated through mutation and recombination genetic operators (with respective probabilities p_m, p_c). The current deme population is then replaced with its offspring together with k 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 $l = (deme_i * m_f)$ fittest individuals of $deme_i$ replace the l worst individuals in the next deme (defined by m_d). Migrants are duplicated, not removed from the source population.

The algorithm iterates until some stopping criteria has been reached. 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.

E. Custom Extensions

Initial tests with 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, k 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 quickly 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[2]. Therefore an extinction prevention (EP) algorithm was introduced which ensures that no model is allowed to go completely extinct (a minimum of 2 individuals per type is enforced). For more information concerning the workings and performance advantages of EP the reader is directed to [2].

Secondly, the attentive reader will have noticed that there remains one major problem with the implementation as discussed so far. 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. The solution we have employed here is to use ensembles (behavioral recombination). If two models of different types are selected to recombine, an average ensemble is created with the models as ensemble members. In addition we enforce a maximum ensemble size ES_{max} and require that ensemble members must differ ES_{diff} percent in their response. Besides enabling hybrid solutions, using ensembles has the additional benefit of allowing a particular model type to lie dormant in an ensemble with the possibility of re-emerging later. Again, for more details see [2].

[2]As an example, this observation was often made when using rational models on EM data.

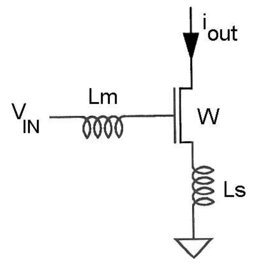

Fig. 2. Narrowband LNA

V. Test Problem

We now compare the performance of the Heterogeneous GA (HGA) described previously to a number of different 'one-shot' approaches. The idea is to check whether the accuracy and type of the final model selected through heterogeneous evolution agrees with the results of a set of independent, single-type, modeling runs.

We consider a test case from electronics: a simple RF circuit, a narrowband Low Noise Amplifier (LNA) [40]. A LNA is the typical first stage of a receiver, having the main function of providing the gain needed to win the noise of subsequent stages, such as a mixer. In addition it has to give negligible distortion to the signal while adding as little noise as possible.

The performance figures of a LNA (gain, input impedance, noise figure and power consumption) can be determined by means of computer simulations where the underlying physics is accurately taken into account. Each simulation typically requires a couple of minutes, too long for a circuit designer to work with efficiently. Instead a designer could use a very accurate surrogate model (based on circuit simulations) to quickly explore how performance figures of LNA scale with key circuit-design parameters, such as the dimensions of transistors, passive components, signal properties and bias conditions.

In this work, in order to keep the computation times manageable, we replace the expensive simulations by a first-order analytic model. As a next step we shall apply the insights gathered in this preliminary phase to the true simulator (where the different one-shot runs are no longer needed, nor possible).

The schematic of the LNA is depicted in figure 2 and the functions that govern its approximate behavior are shown below. The input parameters are the inductances L_m, L_s and the MOSFET width W. The outputs are the admittances y and the noise parameters $\sqrt{\overline{i_{in}^2}}$ and $\sqrt{\overline{i_{out}^2}}$. The approximated y_{11} admittance is defined as

$$y_{11} \cong \frac{j\omega C'_{gs}W}{1-\omega^2 C'_{gs}W(L_s+L_m)+j\omega L_s g'_m W} \quad (1)$$

The other admittance functions are omitted since they are not used in this paper. The approximate input noise-current ($\sqrt{\overline{i_{in}^2}}$) and output noise-current ($\sqrt{\overline{i_{out}^2}}$) are given by equations 2 to 8.

$$f_{gs,in} = \frac{1+j\omega L_s g'_m W}{1-\omega^2 C'_{gs}W(L_s+L_m)+j\omega L_s g'_m W} \quad (2)$$

$$f_{gs,out} = \frac{-j\omega(L_s+L_m)g'_m W}{1-\omega^2 C'_{gs}W(L_s+L_m)+j\omega L_s g'_m W} \quad (3)$$

$$f_{ds,in} = \frac{\omega^2 C'_{gs}W(L_s+L_m)}{1-\omega^2 C'_{gs}W(L_s+L_m)+j\omega L_s g'_m W} \quad (4)$$

$$f_{ds,out} = \frac{1-\omega^2 C'_{gs}W(L_s+L_m)}{1-\omega^2 C'_{gs}W(L_s+L_m)+j\omega L_s g'_m W} \quad (5)$$

$$\overline{i_{gs}^2} = W \cdot \overline{i_{gs}^2}' \text{ and } \overline{i_{ds}^2} = W \cdot \overline{i_{ds}^2}' \quad (6)$$

The remaining parameters have been set to fixed, typical values: $C'_{gs} = 1 \cdot 10^{-9}$ Fm^{-1}, g'_m =100 AV$^{-1}m^{-1}$, $\omega = 2\pi \cdot 5 \cdot 10^9$ Hz, $\overline{i_{gs}^2}' = 2 \cdot 10^4$ pA^{2}Hz^{-1}m^{-1}, $\overline{i_{ds}^2}' = 5 \cdot 10^6$ pA^{2}Hz^{-1}m^{-1}. The meaning of the parameters are as follows: WC'_{gs} is the gate-source capacitance of the MOSFET, Wg'_m is the transconductance of the MOSFET, ω is the angular signal frequency, $\overline{i_{gs}^2}'$ is the gate-source noise current spectral density of the MOSFET, and $\overline{i_{ds}^2}'$ is the drain source noise current spectral density of the MOSFET.

The goal is to create a surrogate model (using our HGA) for this problem using as little simulation points as possible. We hope to see the HGA perform comparable with a number of single, independent runs. Also we 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 optimization landscape) a different species may become dominant. This should result in clearly noticeable population dynamics.

VI. Experimental setup

A. Modeling settings

For the experiments the following model types are used: rational functions (RF), RBF models, RBF SVMs [41], RBF LS-SVMs [42], Artificial Neural Networks (ANN) [43] and Kriging models [44]. Note that for this test case we chose to use the C-based ANN implementation from [43] instead of our own implementation based on the Matlab Neural Network toolbox. Although the Matlab implementation performs at least one order of magnitude better for this test problem, its computational cost is considerably higher.

In addition, the following model parameter optimization algorithms were used: stochastic hill climbing (HC), pattern search (PS), BFGS, and GA. Not all possible combinations were run, only those which experience showed gave good results. The model types included in the heterogeneous GA were: RF, RBF, SVM, LS-SVM and ANN. For the complex-valued y_{11} output only the model types supporting complex data directly were used (RBF and RF). As stated in subsection IV-E the result of a heterogeneous recombination will be an ensemble. So in total up to six model types will be in the running for approximating the data. An in depth treatment of the representation and the genetic operators for each model type is out of scope for this paper but can be found in [2].

B. Sampling settings

An initial optimized Latin hypercube experimental design of size 50, augmented with the corner points, is used. Each iteration 25 new samples are selected using the *gradient* adaptive sampling algorithm, up to a maximum of 500. The *gradient* algorithm is a state-of-the-art sampling method whose working was inspired by existing research into ray tracing. Each iteration it attempts to find an optimal trade-off between error-based and density-based sampling. More information can be found in [45].

C. GA settings

The GA is run for a maximum of 10 generations (though remember that the HGA is restarted with the last solution between each sampling iteration). It terminates if the maximum number of generations is reached, or 4 generations complete with no improvement. The size of each deme is also set to 10. The migration interval m_i is set to 3, the migration fraction m_f to 0.1 and the migration direction is *forward* (copies of the $m_f * popSize$ best individuals from island i replace the worst individuals in island $i+1$). A stochastic uniform selection function was used. The fitness of an individual was defined as the root mean square error (RMSE) on a dynamic validation set of 20%. This means that the validation set grows with the number of available samples, taking care that the distribution of the validation samples stays representative. Though it might not lead to the highest possible accuracy, a validation set is used since it is less expensive (but more biased) than bootstrapping or k-fold cross-validation which are typically used. The remaining parameters are set as follows: $p_m = 0.2, p_c = 0.7, k = 1, p_{swap} = 0.8, ES_{diff} = 10, ES_{max} = 3$ and $EP = true$.

VII. Results

In total 63 experiments were run with each experiment repeated twice (four times for the HGA runs) (the number of repetitions limited by computational constraints). This resulted in a total of 136 runs, each run lasting on average roughly 5-6 hours. The the final averaged RMSE of each run are shown in table I. The ranges of the different outputs are as follows: $y_{11-real}$: [7.75e-05, 6.32e-02], $y_{11-imag}$: [-0.031, 0.031], $\sqrt{\overline{i_{in}^2}}$: [2.51, 563] and $\ln(\sqrt{\overline{i_{out}^2}})$: [50, 54]. The

$$\sqrt{\overline{i_{in}^2}} = \sqrt{|f_{gs,in}|^2 \cdot \overline{i_{gs}^2} + |f_{ds,in}|^2 \cdot \overline{i_{ds}^2} - 2 \cdot \mathrm{Im}(0.4 f_{gs,in} f_{ds,in}^*) \sqrt{\overline{i_{gs}^2} \cdot \overline{i_{ds}^2}}} \quad (7)$$

$$\sqrt{\overline{i_{out}^2}} = \sqrt{|f_{gs,out}|^2 \cdot \overline{i_{gs}^2} + |f_{ds,out}|^2 \cdot \overline{i_{ds}^2} - 2 \cdot \mathrm{Im}(0.4 f_{gs,out} f_{ds,out}^*) \sqrt{\overline{i_{gs}^2} \cdot \overline{i_{ds}^2}}} \quad (8)$$

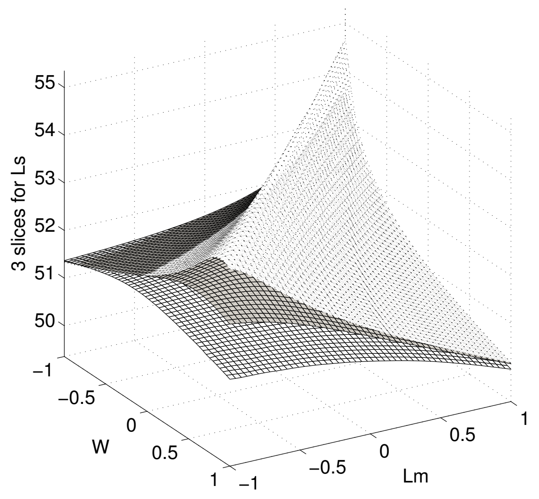

Fig. 3. Plot of final model for $\ln(\sqrt{\overline{i_{out}^2}})$ (normalized, 3 slices for Ls)

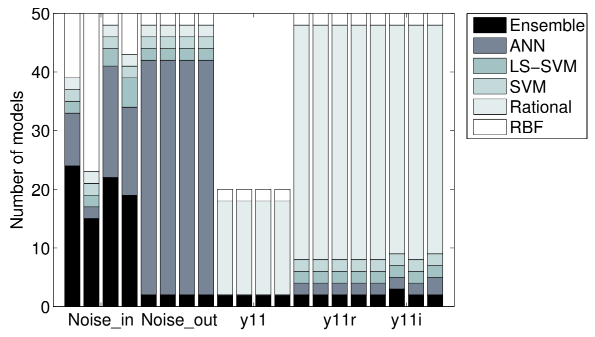

Fig. 4. Composition of the final population

Model	y_{11}	$y_{11-real}$	$y_{11-imag}$	$\sqrt{\overline{i_{in}^2}}$	$\ln(\sqrt{\overline{i_{out}^2}})$
ANN$_{GA}$	-	0.001	0.001	85.697	**0.017**
Kriging$_{GA}$	-	0.007	0.005	58.051	0.536
Kriging$_{PS}$	-	0.007	0.005	52.772	0.576
LS-SVM$_{GA}$	-	0.006	0.004	42.802	0.204
LS-SVM$_{BFGS}$	-	0.007	0.007	34.398	0.207
LS-SVM$_{PS}$	-	0.007	0.005	43.874	0.186
SVM$_{GA}$	-	0.006	0.004	44.982	0.179
SVM$_{BFGS}$	-	0.008	0.005	62.480	0.262
SVM$_{PS}$	-	0.007	0.004	43.494	0.226
Rational$_{HC}$	0.007	0.019	0.062	55.255	0.744
Rational$_{GA}$	1.04e-8	**8.10e-9**	0.004	51.740	0.415
RBF$_{HC}$	0.007	0.006	0.003	21.828	0.180
RBF$_{GA}$	0.006	0.007	0.003	26.506	0.087
HGA	**9.77e-9**	4.51e-8	**2.89e-8**	**15.286**	0.023

TABLE I
FINAL MODEL SCORES (RMSE ON A DYNAMIC VALIDATION SET)

logarithm was taken of the output noise function for numerical reasons (the original range was [5.36e21, 5.63e23]). The composition of the final population for the HGA runs is shown in figure 4. An illustration of the population dynamics during evolution is given in figure 5 (for clarity not all generations are shown). The evolution of the ensemble composition is illustrated in figure 6 (using the same run for $\sqrt{\overline{i_{in}^2}}$ as figure 5). A plot of the final model for $\ln(\sqrt{\overline{i_{out}^2}})$ is shown in figure 3. The form of y_{11} and $\sqrt{\overline{i_{in}^2}}$ are similar.

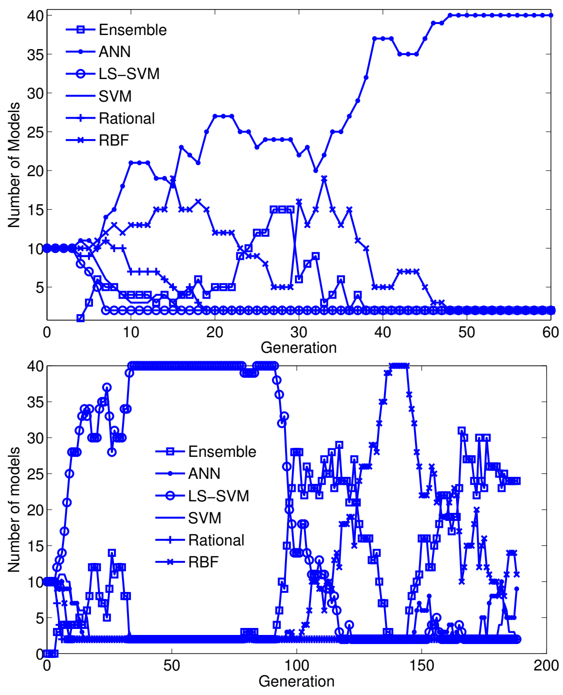

Fig. 5. Population evolution for $\ln(\sqrt{\overline{i_{out}^2}})$ (*top*) and $\sqrt{\overline{i_{in}^2}}$ (*bottom*)

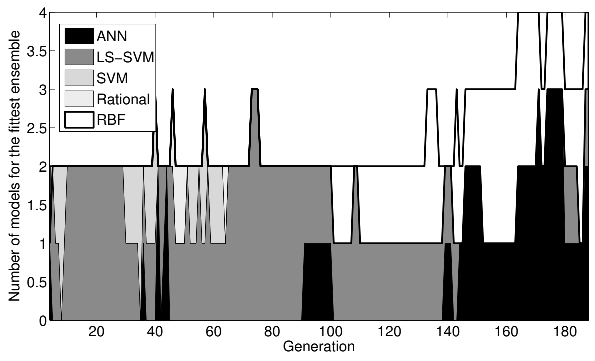

Fig. 6. Ensemble composition evolution for $\sqrt{i_{in}^2}$

VIII. Discussion

Judging from the final scores in table I we see that the admittance y_{11} (and its real/imaginary parts) can be modeled very accurately with rational functions. The noise functions seem somewhat more difficult, with ANN and RBF doing the best on the output noise and RBF on the input noise[3]. These accuracies were confirmed visually and using a pre-calculated test set. If we compare with the results of the HGA we see that the accuracy is comparable or even better in some cases. Since the population size per model type can grow dynamically, the parameter space is searched more extensively, reducing the chance of converging to a poor local optimum (as was the case for one of the rational GA runs for $y_{11-imag}$). If we then compare the best performing model types from table I to the final population composition in figure 4 we see that the correlation is very good. The results for $\sqrt{i_{in}^2}$ are the most interesting. It turns out that in 3 of the 4 runs the best model type was an ensemble of RBFs, ANNs, and occasionally a LS-SVM (figure 6 illustrates the ensemble composition evolution). This implies that the HGA is unable to find a single model type solution that is significantly better than the other model types (which agrees with table I). As a result there is a higher chance of hybrid solutions doing well. This agrees with previous findings, increasing the running time (= adding more data) usually resolves the ambiguity (assuming the data can be accurately fitted by one of the model types). This brings us to a natural strength of our HGA: the best model type need not stay constant, but may change as more information becomes available. All this, together with the usefulness of EP, is nicely illustrated in the top plot of figure 5. Finally, if we compare the performance of the different optimization algorithms, we can only say that the HC algorithm performs very poor (as expected due to it being a more local search). The differences between the other algorithms are marginal, more repetitions and different test cases are needed in order to come to conclusive results. This will be tackled in future work.

[3]If we had included our own ANN implementation it would have performed the best on the noise outputs, achieving 1 order of magnitude lower error.

In sum the results for this test case are promising and in line with previous findings. Also, the advantage over simply running multiple model types in parallel and performing model selection manually is obvious. Such a brute force approach is inelegant, inefficient, lacks any kind of adaptivity or dynamics and does not allow for hybrid solutions like ensembles. We find our heterogeneous approach augmented with EP (to increase robustness) to be a good compromise. An important next step is to study the impact of the different speciation parameters since they determine how the different model types interact. The migration interval is particularly important: if 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.

IX. Conclusion

Due to the computational complexity of current simulation codes, the use of global surrogate modeling techniques has become standard practice among scientists and engineers alike. However, a recurring problem is selecting the most adequate surrogate model type and surrogate model parameter optimization method. In this contribution we have presented a novel approach to tackling these two problems together using the evolutionary migration model. Favorable performance was demonstrated on a case study from electronics.

Future work will consist of exploring the role of the different GA parameters (migration frequency, migration topology, ...) in order to optimize the exploration-exploitation trade-off, incorporating more model types, studying the impact of noisy data, and more advanced ensemble methods (e.g., negative correlation learning). The final goal is to come to a completely adaptive, autonomous, robust model type selection algorithm without having to perform countless, expensive, independent modeling experiments first. Finally it should be noted that all the algorithms and examples described here are available for academic use at `http://sumo.intec.ugent.be`

Acknowledgments

This work was supported by the Fund for Scientific Research Flanders (FWO Vlaanderen), the Science and Innovation Administration Flanders (BSTC project), and the O-MOORE-NICE! project as supported by the European Commission through the Marie Curie program under contract number MTKI-CT-2006-042477.

References

[1] G. G. Wang and S. Shan, "Review of metamodeling techniques in support of engineering design optimization," *Journal of Mechanical Design*, vol. 129, no. 4, pp. 370–380, 2007.
[2] D. Gorissen, "Heterogeneous evolution of surrogate models," Master's thesis, Katholieke Universiteit Leuven (KUL), 2007.
[3] P. K. Davis and J. H. Bigelow, *Motivated Metamodels: Synthesis of Cause-Effect Reasoning and Statistical Metamodeling*. Rand Corporation, The, 2003.

[4] S. D. Mohaghegh, A. Modavi, H. Hafez, M. Haajizadeh, M. Kenawy, and S. Guruswamy, “Development of surrogate reservoir models (srm) for fast track analysis of complex reservoirs,” in *SPE Intelligent Energy Conference and Exhibition. 11-13 April 2006, Amsterdam*, 2006.

[5] R. B. Gramacy, H. K. H. Lee, and W. G. Macready, “Parameter space exploration with gaussian process trees,” in *ICML ’04: Proceedings of the twenty-first international conference on Machine learning*. New York, NY, USA: ACM Press, 2004, p. 45.

[6] Z. Qian, C. C. Seepersad, V. R. Joseph, J. K. Allen, and C. F. J. Wu, “Building surrogate models based on detailed and approximate simulations,” *Journal of Mechanical Design*, vol. 128, no. 4, pp. 668–677, July 2006.

[7] R. R. Barton, “Design of experiments for fitting subsystem metamodels,” in *WSC ’97: Proceedings of the 29th conference on Winter simulation*. New York, NY, USA: ACM Press, 1997, pp. 303–310.

[8] R. Tanese, “Distributed genetic algorithms,” in *Proceedings of the 3rd International Conference on Genetic Algorithms*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1989, pp. 434–439.

[9] M. Nowostawski and R. Poli, “Parallel genetic algorithm taxonomy,” in *Knowledge-Based Intelligent Information Engineering Systems, 1999. Third International Conference*, 31 Aug.-1 Sept. 1999, pp. 88–92.

[10] E. Alba and M. Tomassini, “Parallelism and evolutionary algorithms.” *IEEE Trans. Evolutionary Computation*, vol. 6, no. 5, pp. 443–462, 2002.

[11] G. Folino, C. Pizzuti, and G. Spezzano, “A scalable cellular implementation of parallel genetic programming.” *IEEE Trans. Evolutionary Computation*, vol. 7, no. 1, pp. 37–53, 2003.

[12] 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*, vol. 7, no. 1, pp. 33–47, 1999.

[13] V. S. Gordon, K. Mathias, and D. Whitley, “Cellular genetic algorithms as function optimizers: locality effects,” in *SAC ’94: Proceedings of the 1994 ACM symposium on Applied computing*. New York, NY, USA: ACM Press, 1994, pp. 237–241.

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

[15] C. Hocaoglu and A. C. Sanderson, “Planning multiple paths with evolutionary speciation.” *IEEE Trans. Evolutionary Computation*, vol. 5, no. 3, pp. 169–191, 2001.

[16] 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.

[17] M. Buonanno and D. Mavris, “Aerospace vehicle concept selection using parallel, variable fidelity genetic algorithms,” in *Proceedings of 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Albany, New York*, 2004.

[18] Z. Konfrst, “Parallel genetic algorithms: advances, computing trends, applications and perspectives,” in *Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International*, 26-30 April 2004, p. 162.

[19] 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*, 10-13 May 2005, pp. 111–114.

[20] S. Srinivasulu and A. Jain, “A comparative analysis of training methods for artificial neural network rainfall-runoff models.” *Appl. Soft Comput.*, vol. 6, no. 3, pp. 295–306, 2006.

[21] P. Triverio, S. Grivet-Talocia, M. Nakhla, F. G. Canavero, and R. Achar, “Stability, causality, and passivity in electrical interconnect models,” *IEEE Transactions on Advanced Packaging : Accepted for future publication*, vol. PP, no. 99, pp. 1–1, 2007.

[22] T. W. Simpson, J. D. Poplinski, P. N. Koch, and J. K. Allen, “Metamodels for computer-based engineering design: Survey and recommendations.” *Eng. Comput. (Lond.)*, vol. 17, no. 2, pp. 129–150, 2001.

[23] R. Jin, W. Chen, and T. Simpson, “Comparative studies of metamodelling techniques under multiple modelling criteria,” *Structural and Multidisciplinary Optimization*, vol. 23, no. 1, pp. 1–13, December 2001.

[24] N. V. Queipo, R. T. Haftka, W. Shyy, T. Goel, R. Vaidyanathan, and P. K. Tucker, “Surrogate-based analysis and optimization,” *Progress in Aerospace Sciences*, vol. 41, pp. 1–28, 2005.

[25] V. C. Chen, K.-L. Tsui, R. R. Barton, and M. Meckesheimer, “A review on design, modeling and applications of computer experiments,” *IIE Transactions*, vol. 38, pp. 273–291, 2006.

[26] M. Streeter and L. A. Becker, “Automated discovery of numerical approximation formulae via genetic programming,” *Genetic Programming and Evolvable Machines*, vol. 4, no. 3, pp. 255–286, 2003.

[27] Y.-S. Yeun, W.-S. Ruy, Y.-S. Yang, and N.-J. Kim, “Implementing linear models in genetic programming.” *IEEE Trans. Evolutionary Computation*, vol. 8, no. 6, pp. 542–566, 2004.

[28] Y.-K. Kwon, B.-R. Moon, and S.-D. Hong, “Critical heat flux function approximation using genetic algorithms,” *Nuclear Science, IEEE Transactions on*, vol. 52, no. 2, pp. 535–545, April 2005.

[29] S. Lessmann, R. Stahlbock, and S. Crone, “Genetic algorithms for support vector machine model selection,” in *Neural Networks, 2006. IJCNN ’06. International Joint Conference on*, 16-21 July 2006, pp. 3063–3069.

[30] S. Tomioka, S. Nisiyama, and T. Enoto, “Nonlinear least square regression by adaptive domain method with multiple genetic algorithms,” *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 1, pp. 1–16, February 2007.

[31] X. Yao, “Evolving artificial neural networks,” *Proceedings of the IEEE*, vol. 87, no. 9, pp. 1423–1447, Sept. 1999.

[32] Y. Jin, M. Olhofer, and B. Sendhoff, “A framework for evolutionary optimization with approximate fitness functions.” *IEEE Trans. Evolutionary Computation*, vol. 6, no. 5, pp. 481–494, 2002.

[33] I. Paenke, J. Branke, and Y. Jin, “Efficient search for robust solutions by means of evolutionary algorithms and fitness approximation.” *IEEE Trans. Evolutionary Computation*, vol. 10, no. 4, pp. 405–420, 2006.

[34] Y.-S. Ong, P. Nair, and K. Lum, “Max-min surrogate-assisted evolutionary algorithm for robust design,” *Evolutionary Computation, IEEE Transactions on*, vol. 10, no. 4, pp. 392–404, Aug. 2006.

[35] K. O. Stanley and R. Miikkulainen, “Evolving neural networks through augmenting topologies,” *Evol. Comput.*, vol. 10, no. 2, pp. 99–127, 2002.

[36] D. Gorissen, W. Hendrickx, K. Crombecq, and T. Dhaene, “Adaptive distributed metamodeling,” in *Post conferenc proceedings of 7th International Meeting on High Performance Computing for Computational Science (VECPAR 2006), Springer - Lecture Notes in Computer Science*, M. D. et al., Ed., vol. LNCS 4395, 2007, pp. 579–588.

[37] M. Ding and R. Vemur, “An active learning scheme using support vector machines for analog circuit feasibility classification,” in *18th International Conference on VLSI Design*, 3-7 Jan. 2005, pp. 528–534.

[38] T. G. Robertazzi and S. C. Schwartz, “An accelerated sequential algorithm for producing d-optimal designs,” *Siam Journal on scientific Computing*, vol. 10, pp. 341–358, 1989.

[39] A. C. Keys and L. P. Rees, “A sequential-design metamodeling strategy for simulation optimization,” *Comput. Oper. Res.*, vol. 31, no. 11, pp. 1911–1932, 2004.

[40] T. H. Lee, *The Design of CMOS Radio-Frequency Integrated Circuits*, 2nd ed. Cambridge University Press, 2004.

[41] C.-C. Chang and C.-J. Lin, *LIBSVM: a library for support vector machines*, 2001, software available at http://www.csie.ntu.edu.tw/c̃jlin/libsvm.

[42] J. Suykens, T. V. Gestel, J. D. Brabanter, B. de Moor, and J. Vandewalle, *Least Squares Support Vector Machines*. Singapore: World Scientific Publishing Co., Pte, Ltd., 2002.

[43] M. Nørgaard, O. Ravn, L. Hansen, and N. Poulsen, “The NNSYSID toolbox,” in *IEEE International Symposium on Computer-Aided Control Sysstems Design (CACSD), Dearborn, Michigan, USA*, 1996, pp. 374–379. [Online]. Available: http://www2.imm.dtu.dk/pubdb/p.php?2733

[44] H. Lophaven, S. Nielsen, and J. Søndergaard, “DACE - a MATLAB Kriging toolbox,” http://www.imm.dtu.dk/~hbn/dace/.

[45] K. Crombecq, “A gradient based approach to adaptive metamodeling,” University of Antwerp, Tech. Rep., 2007.