Evolutionary Neuro-Space Mapping Technique for Modeling of Nonlinear Microwave Devices

Dirk Gorissen, *Student Member, IEEE*, Lei Zhang, *Member, IEEE*,
Qi-Jun Zhang, *Fellow, IEEE*, and Tom Dhaene, *Senior Member, IEEE*

***Abstract*—This paper presents a new advance in Neuro-Space Mapping (Neuro-SM) techniques for modeling nonlinear microwave devices. Suppose that existing device models (namely, coarse models) cannot match the behavior of a new device (referred to as the fine model). By neural network mapping of the voltage and current signals from the coarse to the fine models, Neuro-SM can modify the behavior of the coarse model to match that of the fine model. However, the efficiency of mapping depends on both the mapping structure and the coarse model. In the present paper, a structural optimization technique is presented to achieve optimal combinations of mapping structure and coarse model. An aggressive optimization formulation exploring detailed structural variations in both the mapping and the coarse model is proposed, where the internal branches of coarse models and separate mappings for the voltage and current at gate and drain are used as basic topology variables. The formulation of such a structural optimization by an evolutionary optimization algorithm is proposed. Numerical examples of Metal Semiconductor Field Effect Transistor and High Electron Mobility Transistor modeling demonstrate that, by using the proposed algorithm, optimal combinations of space mapping and coarse model structures can be achieved leading to the best modeling accuracy with the simplest mapping function.**

***Index Terms*—genetic algorithms, knowledge-based modeling, nonlinear device modeling, space mapping**

I. Introduction

Modeling and computer-aided design (CAD) techniques are important in helping microwave designers to achieve efficient design of linear and nonlinear microwave circuits. In recent years, artificial neural networks (ANNs) [1]–[7] and space mapping [8]–[12] have been recognized as two important developments in microwave CAD to address the growing computational challenges in modeling, simulation and optimization. Techniques combining ANNs and space mapping have been developed for electromagnetic modeling [2], nonlinear device modeling [13] and statistical device modeling [14]. In addition to the combination of space mapping with ANNs, implementations using Linear Regression Models [15], Neuro-Fuzzy models [16], locally weighted models [17], and hybrid models [18], have also been described.

T. Dhaene is with the Department of Information Technology (INTEC), Ghent University-IBBT, Gaston Crommenlaan 8, 9050 Gent, Belgium (email: tom.dhaene@ugent.be). D. Gorissen was with INTEC, Ghent University-IBBT and is now with the Computational Engineering and Design Group, Southampton University, University Way 1, SO17 1BJ, UK (email: dirk.gorissen@soton.ac.uk). L. Zhang was with the Department of Electronics, Carleton University, and is now with the RF Division, Freescale Semiconductor, 2100 E. Elliot Drive, Tempe, Arizona, 85284, USA (email: lei.zhang@freescale.com). Q.J. Zhang is with the Department of Electronics, Carleton University, 1125 Colonel By Drive, Ottawa, ON, K1S 5B6, Canada (email: qjz@doe.carleton.ca)

This paper explores further advances in the application of ANN and space mapping for modeling of nonlinear microwave devices. Nonlinear device modeling is an important area of microwave CAD, and many device models have been developed [19], [20], such as physics-based models [21]–[23], equivalent circuit based models [24]–[29], or table based models [30]. With the continuous development of semiconductor device technologies, new devices constantly evolve and existing models need to be updated. The need for faster model development cycles also demands new CAD methods for modeling, so the task of model development becomes more efficient and systematic.

Recently, a CAD method for nonlinear device modeling, called Neuro-Space mapping (Neuro-SM) technique has been introduced [13], [31]. The methods start from a known equivalent circuit model that is already a coarse approximation of the new device behavior. We refer to this existing equivalent circuit model as the coarse model. A generic method like ANN is then applied to map or modify the voltage/current relationship in the coarse model to match that of the new device data. The final model, i.e., the Neuro-SM model, is a combination of both the ANN mapping and the coarse model. The ANN part of the overall model is referred to as the mapping structure, and the existing equivalent circuit model as the knowledge that is integrated with the ANN. In this way we can "repair" (i.e., improve) an existing model by a CAD method such that the final model matches the new device much better. The specific mapping structure for nonlinear device modeling introduced in [31] and [13] was based on an input-mapping concept, the earliest type of space mapping originally formulated for EM optimization [10]. There is no guarantee that the input-mapping structure is the best mapping structure for all device examples. For example, the input mapping may not be sufficient if the output of the fine model is beyond the output range of the coarse model. Fortunately, a variety of mapping methods have been developed in passive/EM modeling that can be adopted for nonlinear device modeling, such as space mapping [8], [11]–[14], difference mapping (or difference method) [5], output mapping [5], prior knowledge input (PKI) method [32], and knowledge-based neural network involving hybrid mapping [5], [33].

The overall efficiency of a general Neuro-space mapping model depends on the quality of the equivalent circuit model and the suitability of the mapping structure. In order to obtain optimal overall model accuracy it is important that the best combination of equivalent circuit model and mapping structure is used. Which model/mapping combination performs best is

difficult to determine up-front since it depends on the data and the problem characteristics [11], [33]. Thus, this problem must be solved on a case-by-case basis, each time a new device is considered. However, there are many combinations of possible equivalent circuit models and mapping structures. It is very expensive to fully exploit this rich combination due to the cumbersome manual process of selecting structural combinations and optimizing the model and mapping parameters exhaustively. This also means that there is still a large potential for improvements in accuracy if the space of different combinations is searched more efficiently. This brings us to the motivation of this paper.

We present a new methodology for nonlinear device modeling that is not tied to a particular device, an empirical approximation, or a mapping structure. The paper describes an approach to explore the space of mapping structures by formulating the Neuro-SM structural optimization problem with an evolutionary optimization algorithm. To enrich the optimization search space and extract maximum possible improvements in modeling with simpliest mapping function, we also break down the mapping structure into finer structural variables for optimization. For example, separate mapping structures for voltage mapping, current mapping, transistor gate mapping, or drain mapping. In addition, since the efficiency of the mapping depends on the quality of the nonlinear equivalent circuit model, we go one level deeper by decomposing the equivalent circuit models into their constituent internal branches and allow mixing of these branches across different model types. This further enriches the optimization search space, i.e., the different combinations of mapping structures and equivalent circuit models, allowing potentially superior models compared to traditional pure equivalent circuit models or hybrid models with predetermined mapping structures such as in [13]. This also allows for simpler mapping functions, which is desirable. In our implementation of the optimization, the final solution (a model) from the evolutionary optimization program is a simple netlist representing the complete and optimal model structure (including equivalent circuit and mapping network), and optimized parameter values of all the functions in the model.

II. Evolutionary Knowledge-Based Modeling

A. Motivation

The starting point of our work is that the existing model cannot match the new device behavior. The existing equivalent circuit models, called coarse models, need to be modified and extended in order to accommodate for new device behavior. Manual modification of models is a trial and error process and hybrid methods have been developed to help map the coarse model to the device data [8], [11]–[14], [31], [33], [34].

However, there are a variety of mapping methods and the optimal mapping choice is typically not well defined since it depends mutually on the problem and the given nonlinear equivalent circuit model (the better the nonlinear equivalent circuit model, the simpler the mapping structure). This mutual dependence cannot be solved in an a priori manner since no mathematical procedure is available for such variations of space mapping methods during nonlinear device modeling.

A further problem is that with a given set of equivalent models as is, simple mapping functions may still be insufficient to achieve the desired accuracy. We need to allow both parametric and structural variations inside the equivalent circuit models to improve the accuracy with respect to the data, so that we can achieve best accuracy in the final model with simplest mapping functions.

The result is a combinatorial explosion due to the cross-product of different mapping methods and structural variations of each equivalent circuit model type. Navigating this search space is very human intensive and can only be done for a limited set of possible solutions. Significant cost savings and potentially large gains in accuracy could be obtained if this search is automated and parallelized in an efficient way.

B. Genetic Algorithms

We propose the use of evolutionary search to tackle the aforementioned problem. In particular, a genetic algorithm is defined that starts from an initial population of mapping structures and nonlinear equivalent circuits (which may be randomly generated) and uses specific reproduction operators to generate new circuit models based on the previous population. Proper definition of the reproduction operators allows the algorithm to efficiently explore the large search space to quickly come to an optimal solution.

An important advantage of evolutionary methods over classic optimizers such as Broyden–Fletcher–Goldfarb–Shanno is that evolutionary methods are well suited to structural topology optimization while classic methods are more suitable to continuous optimization problems in a real valued space. An advantage over other direct search methods like Particle Swarm Optimization [35] is that the user can exercise fine control over the concrete representation and genetic operators used. This allows the genetic algorithm to be fine-tuned to the problem at hand, making the incorporation of problem specific knowledge, constraints, and requirements more convenient.

C. Chromosomal Encoding of the Model Search Space

The purpose of the genetic algorithm is to find the optimal mapping structure for an existing coarse model. The associated topology of the equivalent circuit model, or coarse model, can also be optimized automatically to explore the search space in order to come to an improved overall model.

In order to make this possible, we must first formulate a coding scheme to represent the different mappings and associated equivalent circuits. This coding scheme must be a function that uniquely translates a mapping structure/equivalent circuit model combination into a code that the genetic algorithm can work with. For this paper, we take the building blocks of the code to be the different mapping types and the different elements (different branches) of one or more existing equivalent circuit models.

In the next subsection we first discuss the base equivalent circuit model employed in this paper and how the different types of mapping structures are implemented around it. The subsequent subsection will then discuss how the equivalent circuit model itself can be encoded and its topology optimized.

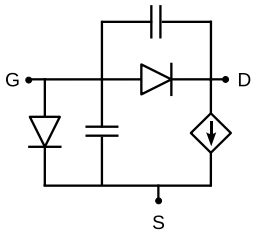


Figure 1: Base circuit model example (the intrinsic equivalent circuit of a transistor) with three nodes (Gate, Drain, and Source). All branches, including the capacitors, are nonlinear elements in general.

1) Mapping Structure Implementation: This paper is concerned with modeling a transistor and we focus on the most difficult part of the transistor modeling, the nonlinear intrinsic part [36]. Assuming that the extrinsic elements are fixed and can be separately determined from *S*-parameter measurements [36], we only consider modeling the intrinsic circuit. We use a base circuit model with three nodes for the intrinsic modeling of the transistor device, i.e., the gate, the drain, and the source terminals, denoted by $G, D,$ and S, respectively. We denote the branches connecting the three nodes by $\mathbf{B} = \{GS, GD, DS\}$, with each branch containing one or more elements (e.g., diode, nonlinear capacitor, nonlinear controlled current source, etc.). An example of such a circuit structure is shown in Fig. 1. This is an example of the coarse model part in the overall model.

We now temporarily assume a fixed topology for the equivalent circuit model and discuss the different mapping structures and how they are implemented. With mapping we mean that the gate and drain voltage signals of the device (i.e., the fine model) will not be applied directly to the coarse model. Instead, they will be modified (mapped) and only then applied to the coarse model. Similarly, the gate and drain current signals can also be mapped. However, there are many different ways in which the behavior of the coarse model may be misaligned with the true device data: Ranging from a simple misalignment easily corrected by a shift and scaling (linear mapping) in the input space, to a highly nonlinear misalignment requiring a complex correction operator in the output space. Depending on the nature of the misalignment (simple, complex, in the input space or output space, etc.) different mapping methods or a combination of methods will be required. It is not always obvious which mapping method is most suited in advance.

Fig. 2 shows the base circuit model example from Fig. 1 extended with elements that make the different mappings possible. A common source implementation is considered here. The interpretation is as follows: given the gate and drain terminal voltage signals v_g and v_d as inputs, the gate and drain current signals i_g and i_d of the modeling problem are solved from those of the equivalent circuit model (defined as $v_{g_{map}}, v_{d_{map}}, i_{g_{map}},$ and $i_{d_{map}}$) through the application of input mapping, output mapping, difference mapping, or any subset thereof. We now discuss each of these three possibilities in turn.

a) Input Mapping: In the input mapping method, the mapping function maps the input space of the original problem onto a coarse model input space [13]. The coarse model is an existing equivalent circuit model that cannot represent a new device behavior accurately in the original input space. By applying input mapping, the coarse model with mapped inputs can produce the outputs with improved accuracy [13]. Input space mapping is achieved by adding voltage controlled voltage sources that perform a mapping of v_g and v_d onto $v_{g_{map}}$ and $v_{d_{map}}$ through

$$v_{g_{map}} = f_{IM_g}(v_g, v_d) \tag{1}$$

$$v_{d_{map}} = f_{IM_d}(v_g, v_d) \tag{2}$$

where $f_{IM_g}(\cdot)$ and $f_{IM_d}(\cdot)$ represent the input mapping equations for the gate and drain voltage signals, respectively. Input space mapping is most effective when only the input of the coarse model needs to be realigned [33]. The simpler this realignment (i.e., depending on the coarse model) the more straightforward the implementations of $f_{IM_g}(\cdot)$ and $f_{IM_d}(\cdot)$. Thus, a poor choice for the coarse model will make constructing an accurate mapping function considerably more difficult. If no input mapping is requested by the algorithm, an identity function is used, i.e., $f_{IM_g}(v_g, v_d) = v_g$ and $f_{IM_d}(v_g, v_d) = v_d$.

b) Output Mapping: With output mapping, the appoximator learns the relationship between the outputs of a prior knowledge model and the original problem. We use the Prior Knowledge Input (PKI) formulation [32] of output mapping. Output mapping is applied on the output currents and input voltages of the equivalent circuit model using controlled current sources:

$$i'_g = f_{OM_g}(v_g, v_d, i_{g_{map}}) \tag{3}$$

$$i'_d = f_{OM_d}(v_g, v_d, i_{d_{map}}) \tag{4}$$

where i'_g and i'_d are intermediate values of gate and drain currents, respectively, and $f_{OM_g}(\cdot)$ and $f_{OM_d}(\cdot)$ represent the output mapping equations for the gate and drain current signals. Hence, as opposed to input space mapping, output mapping is most effective when only the output of the coarse model needs to be realigned [33]. Similarly, an optimal choice for $f_{OM_g}(\cdot)$ and $f_{OM_d}(\cdot)$ depends on the quality of the coarse model. Again an identity mapping is used if no output mapping is selected by the algorithm, i.e., $f_{OM_g}(\cdot) = i_{g_{map}}$ and $f_{OM_d}(\cdot) = i_{d_{map}}$.

c) Difference Mapping: The mapped output currents from (3) and (4) can further be corrected by applying difference mapping using controlled current and charge elements, in order to obtain the final output currents

$$i_g = i'_g + f_{DM_g}(v_g, v_d) \tag{5}$$

$$i_d = i'_d + f_{DM_d}(v_g, v_d) \tag{6}$$

where

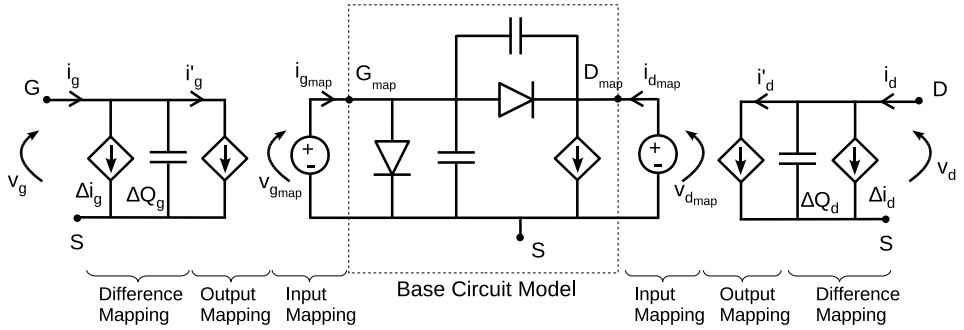


Figure 2: Base circuit model from Fig. 1 extended with mappings (represented by controlled sources in the circuit form). All three mappings (input mapping, output mapping, and difference mapping) are shown on the figure, both on the gate terminal and the drain terminal.

$$f_{DM_g}(v_g, v_d) = \Delta i_g(v_g, v_d) + \Delta \dot{Q}_g(v_g, v_d) \tag{7}$$

$$f_{DM_d}(v_g, v_d) = \Delta i_d(v_g, v_d) + \Delta \dot{Q}_d(v_g, v_d) \tag{8}$$

In (7) and (8), Δi_g (Δi_d) and ΔQ_g (ΔQ_d) represent the current and charge corrections for the gate (drain) terminal, and $\Delta \dot{Q}_g$ and $\Delta \dot{Q}_d$ are the time derivatives of the correction charges. $f_{DM_g}(\cdot)$ and $f_{DM_d}(\cdot)$ represent the difference mapping equations and both are zero if no difference mapping exists. Difference mapping is most effective if the difference between the coarse model and the true data follows a predictable pattern [33].

There are many possible implementations for $f_{IM_g}(\cdot)$, $f_{IM_d}(\cdot)$, $f_{OM_g}(\cdot)$, $f_{OM_d}(\cdot)$, $f_{DM_g}(\cdot)$ and $f_{DM_d}(\cdot)$, ranging from simple linear mappings to powerful ANN-based mappings [13], [14]. As discussed in Subsection II-A, the optimal choice will depend on the problem and available coarse model structures. By allowing topological optimization in coarse models, our algorithm achieves overall model accuracy by maximum use of coarse model information with the simpliest form of mapping, the linear mapping.

2) Equivalent Circuit Encoding and Solution Representation: Recall from Fig. 1 that we assume a base circuit model with 3 nodes denoted by G, D, S and three connecting branches $\mathbf{B} = \{GS, GD, DS\}$. Let $E_{b,b\in\mathbf{B}}$ be the discrete variable representing the type of element present in a branch (e.g., $E = 1$: Diode, $E = 2$: Capacitor, etc). In addition, since we wish to explore different circuit topologies we allow multiple parallel branches for each of the 3 main intrinsic branches. Let p denote the number of such parallel branches. This means that a complete circuit with a maximum of p parallel branches for each main branch can be written as a set of three-tuples:

$$\{(E_{GS_1}, E_{GD_1}, E_{DS_1}), ..., (E_{GS_p}, E_{GD_p}, E_{DS_p})\} \tag{9}$$

This is illustrated graphically in Fig. 3. As an example, the equivalent circuit model shown in Fig. 1 can be expressed by taking $p = 2$, two nonlinear capacitors for E_{GS_1} and E_{GD_2}, two diodes for E_{GS_2} and E_{GD_1}, a nonlinear source for E_{DS_1}, and using the respective formula for each element.

Besides different circuit topologies, we go beyond pure models where each element is approximated by the same type

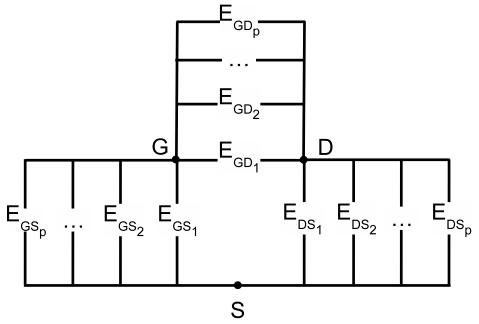


Figure 3: Generalized base circuit model with p parallel branches connecting the base circuit nodes G, D and S.

of equivalent circuit model. We wish to allow a hybrid equivalent circuit model where each element E_{b_r} ($b \in \mathbf{B}, r = 1, .., p$) may be approximated by a different equivalent circuit model type. Let M be a discrete variable that represents the model type (e.g., $M = 1$: Curtice model [24], $M = 2$: Materka model [26], etc.). The circuit representation then becomes:

$$\begin{aligned}\{&((M,E)_{GS_1}, (M,E)_{GD_1}, M_{DS_1}), ..., \\ &((M,E)_{GS_p}, (M,E)_{GD_p}, M_{DS_p})\}\end{aligned} \tag{10}$$

with the tuple $(M,E)_{b_r}$ representing the element E in parallel branch r of main branch $b \in \mathbf{B}$ whose approximation is dictated by model type M. Since we assume the use of voltage controlled current sources for the DS branch, we can omit the element type parameter in the DS branch from the chromosome representation.

Finally, we come to the optimal choice of the mapping structure, which is applied to the circuit model as discussed so far. Let K denote the type of mapping used (e.g., $K = 1$: input mapping, $K = 2$: output mapping, etc.) and let the subscripts G and D indicate if the mapping is applied on the gate terminal or the drain terminal. If a mapping is not applied, this is denoted by $K = 0$. Together, this results in the following final circuit representation in a matrix form:

$$\left[\begin{array}{cc|cc|cc|cc} M_{GS_1} & E_{GS_1} & M_{GD_1} & E_{GD_1} & M_{DS_1} & K_{G_1} & K_{D_1} \\ M_{GS_2} & E_{GS_2} & M_{GD_2} & E_{GD_2} & M_{DS_2} & K_{G_2} & K_{D_2} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ M_{GS_p} & E_{GS_p} & M_{GD_p} & E_{GD_p} & M_{DS_p} & K_{G_p} & K_{D_p} \end{array}\right] \tag{11}$$

Note that this setup allows different mappings to be applied to the gate and drain separately and allows multiple mapping types to occur concurrently. The only restriction is that input mapping, output mapping, and difference mapping may only occur once in each mapping column (for example, applying output mapping twice on the drain terminal is not allowed). Absence of a certain branch or mapping is indicated by zeros at the corresponding locations. A graphical representation of the matrix in (11) is shown in Fig. 4.

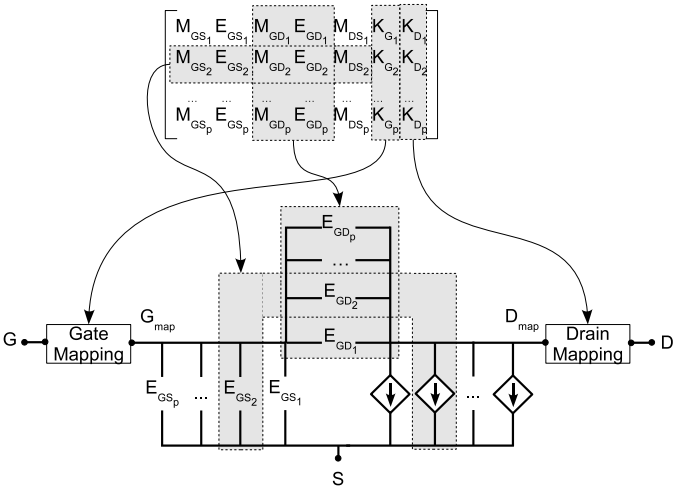


Figure 4: Graphical representation of encoding an equivalent circuit model and mapping structures as a matrix. A mapping type can only occur once in each mapping column. If a branch or mapping is not present this is denoted by zeros at the corresponding locations.

The matrix in equation (11) represents the chromosome of an individual, with M, E, K representing the different genes. In a real organism each gene has a number of alleles that can occupy each of the gene loci. The same is true here. In the concrete implementation, each of the symbols M, E, K is represented by a number that indicates the model type, the element type, and the mapping type, respectively. Table I shows the different alleles used for each gene. Thus, we have reduced the problem to combinatorial optimization, the total number of combinations being:

$$\left[\sum_{l=1}^{p}\binom{(N_M \cdot N_E)+l-1}{l}\right]^2 \cdot \left[\sum_{l=1}^{p}\binom{N_M+l-1}{l}\right] \cdot \left[\sum_{l=0}^{N_K}\binom{N_K}{l}\right]^2 \tag{12}$$

with N_M, N_E and N_K representing the number of different model types, element types, and mapping types, respectively. Thus each circuit generated by the genetic algorithm is encoded as a p-by-7 matrix, where p is the number of parallel branches defined in (9).

Table I: Encoding table containing codes for each allele used in the chromosomal encoding.

	Chromosomal encoding
Model type (M)	1: Curtice, 2: Materka, 3: Chalmers
Element type (E)	1: Capacitor, 2: Diode
Mapping type (K)	1: Input mapping, 2: Output mapping 3: Difference mapping

An example of a coarse model (without mapping structures) and its equivalent representation in matrix form is shown in Fig. 5. The circuit has a Chalmers capacitor and a Curtice diode in the *GS* branch and two Materka capacitors with different nonlinear coefficients, a Curtice diode, and a Chalmers diode in the *GD* branch. A Materka source in the *DS* branch completes the circuit. Such topological mixing of different equivalent circuit models allows for more freedom and flexibility to fit the device behavior with improved accuracy over traditional homogeneous equivalent circuit models and/or to permit a simpler mapping structure.

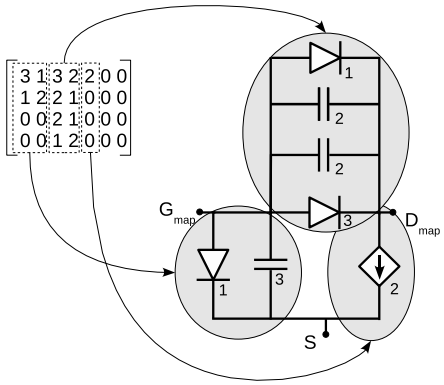


Figure 5: Example of encoding a circuit topology into a chromosome using the proposed encoding from (11). The numbers on the circuit represent the model type used (see Table I). The circuit has a Chalmers capacitor and a Curtice diode in the *GS* branch and two Materka capacitors with different nonlinear coefficients, a Curtice diode, and a Chalmers diode in the *GD* branch. A Materka source in the *DS* branch completes the circuit. Such hybrid of different model types allows flexibility of mapping and expands the search space for improved accuracy.

An example with explicit mapping structures is depicted in Fig. 6. In this example input mapping and output mapping are applied to the gate terminal and difference mapping is applied to the drain terminal.

D. Genetic Operators

Once an encoding scheme has been defined, it becomes possible to define the genetic operators. Recall from Section II-B that the genetic algorithm starts from a population of transistor models. Based on this population (namely the parents), the genetic algorithm generates a new population (namely the children) of transistor models based on two genetic operators: mutation and crossover. Which models are selected as parents

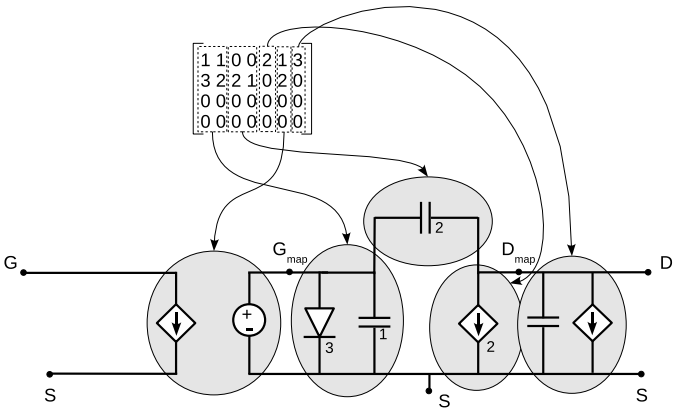


Figure 6: Circuit encoding example with mapping structures. Input and output mappings are applied to the gate terminal, while difference mapping is applied to the drain terminal.

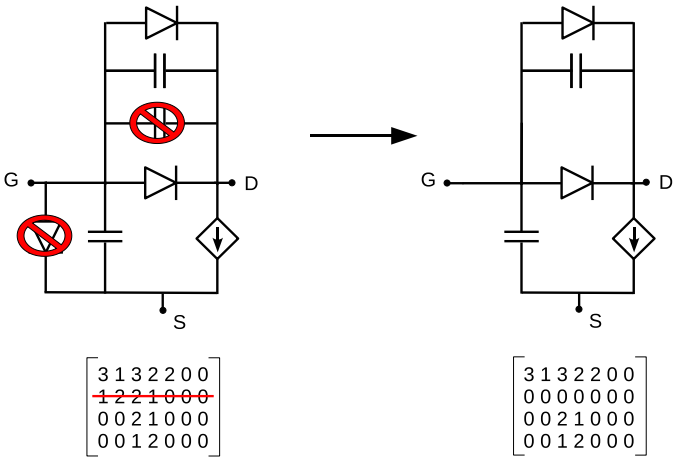


Figure 7: Example of applying the *"Delete a random row"* mutation operator, resulting in eliminating the diode in the *GS* branch and the capacitor in the *GD* branch.

is determined by a selection function. The selection function will take into account the fitness (defined as the model accuracy in terms of training error, described in section II-E) of each transistor model so that models with a better accuracy (or lower training error) have a higher probability of being selected as parents. Thus by iteratively applying the genetic operators on a starting population, selection should drive the population to the optimal solution.

1) Mutation: The purpose of the mutation operator is to ensure that every region of the search space can be reached. It should therefore contain sufficient randomness to make this possible. We employ a meta-mutation operator that selects one or more simple mutation operators according to various probabilities. Let X be the matrix in (11) which represents a particular encoded circuit. The simple mutation operators are formulated as follows:

- *Delete a random row*: a row r $(1 \leq r \leq p)$ is chosen randomly from X and replaced by a row of zeroes. This has the effect of deleting one parallel branch from each of the three main circuit branches (GD, GS, DS). This operator is illustrated in Fig. 7.
- *Replace a random row*: a row r is chosen randomly from X and replaced by a randomly generated vector $[M_{GS_r}, E_{GS_r} M_{GD_r}, E_{GD_r}, M_{DS_r}, K_{G_r}, K_{D_r}]$. This will add or replace an existing mapping structure or parallel branch.
- *Rotate the model types*: the non-zero model types $[M_{GS_r}, M_{GD_r}, M_{DS_r}]$ of each row r of X are permuted one clockwise cycle resulting in $[M_{DS_r}, M_{GS_r}, M_{GD_r}]$. For example, if this operator is applied to the circuit in Fig. 5 the *GS* branch will contain a Materka capacitor (type inherited from the original Materka source in the *DS* branch) and Materka diode (from the Materka capacitor in the second parallel branch of *GD*) instead of a Chalmers capacitor and Curtice diode.
- *Rotate the element types*: the same as the previous operator only now applied to the element types in the *GS* and *GD* branches.
- *Swap the mapping structures*: the mapping structures that are applied to the drain terminal are switched to the gate terminal, and vice versa. Thus K_{D_r} swaps places with K_{G_r} for each row r of X. An illustration is given in Fig. 8. First input mapping and output mapping were applied on the gate terminal and difference mapping on the drain terminal. After applying the mutation operator the situation is reversed.
- *Delete a random branch*: a row r and branch b from X is chosen randomly, uniquely identifying a particular parallel branch, i.e., a tuple (M_{b_r}, E_{b_r}). This branch is then deleted from the circuit. This procedure is repeated three times.
- *Replace a random branch*: the same as the previous operator instead now the randomly selected branch is replaced by a randomly generated one. If the selected branch was empty (consisted of zeroes) this amounts to adding a new parallel branch. This is repeated twice.

The inherent nature of Genetic algorithms is that many aspects of the algorithm are decided using probability rather than deterministically. For example, which subset of operators is applied on a given individual will depend on the outcome of a random number generator. The advantage of this approach is that it still allows for large jumps in the search space without completely destroying the individual in one step (as would be the case if one single, complex mutation operator was used). Note that the collection of simple operators edge on the conservative side when it comes to circuit complexity. Three of the operators leave the size of the circuit unchanged, two of the operators remove branches, while two potentially add new elements. This helps ensure that the genetic algorithm does not needlessly generate overly complex circuits.

2) Crossover: The purpose of the crossover operator is to recombine genetic information from two parents in order to produce one or more offspring. The offspring contain recognizable genetic information from both parents.

We again use a combination of different simple recombination operators that act on two parents X_1 and X_2, both encoded using (11):

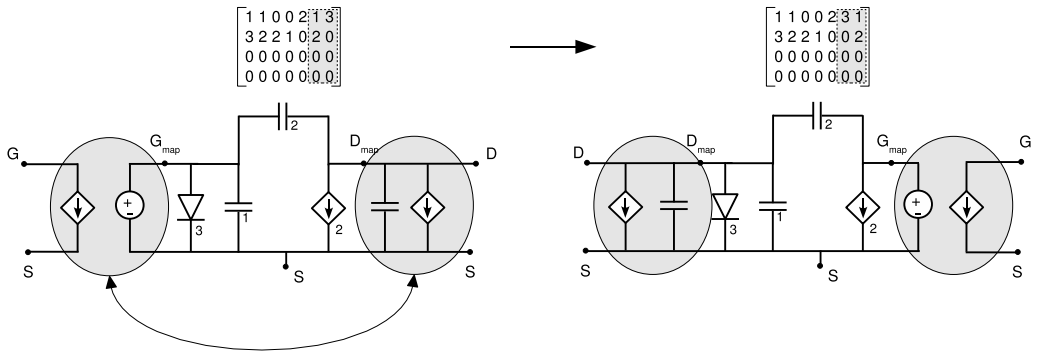


Figure 8: Example of applying the *"Swap the mapping structures"* mutation operator. The mappings that applied to the gate terminal are transferred to the drain terminal and vice versa. In the figure this means that first input mapping and output mapping were applied on the gate and difference mapping on the drain. After applying the mutation operator the situation is reversed.

- *Swap the model types*: the model type sub-matrix of X_1,

$$M_{X_1} = \begin{bmatrix} M_{GS_1} & M_{GD_1} & M_{DS_1} \\ M_{GS_2} & M_{GD_2} & M_{DS_2} \\ \dots & \dots & \dots \\ M_{GS_p} & M_{GD_p} & M_{DS_p} \end{bmatrix}_{X_1} \tag{13}$$

  replaces the equivalent sub-matrix M_{X_2} of X_2 and vice versa. To prevent orphaned model types only non-zero elements are replaced. As an example, this means a parent circuit which contains a mixture of Materka and Curtice elements in its *GD* branch will result in a child circuit with the exact same elements but now approximated by the Chalmers equations (if we assume the other parent only has Chalmers elements in its *GD* branch).
- *Swap the elements*: this operation is equivalent to the previous operator except this time only element type information is exchanged. The relevant sub-matrix $E_{X_{1,2}}$ is then

$$E_{X_{1,2}} = \begin{bmatrix} E_{GS_1} & E_{GD_1} \\ E_{GS_2} & E_{GD_2} \\ \dots & \dots \\ E_{GD_p} & E_{GD_p} \end{bmatrix}_{X_{1,2}} \tag{14}$$

  Thus now the element types are changed while the model types remain fixed. For example, application to a circuit with two Materka diodes in a certain branch may result in a circuit with two Materka capacitors in that branch.
- *Swap the mapping structures*: analogous to the previous operators the mapping structure sub-matrix $K_{X_{1,2}}$ is exchanged

$$K_{X_{1,2}} = \begin{bmatrix} K_{G_1} & K_{D_1} \\ K_{G_2} & K_{D_2} \\ \dots & \dots \\ K_{G_p} & K_{D_p} \end{bmatrix}_{X_{1,2}} \tag{15}$$

  As an example, by applying this operator, the mapping structures that surrounded the equivalent circuit model of X_1 are transferred to the equivalent circuit of X_2. Likewise, the mapping structures that surrounded the equivalent circuit of X_2 are transferred to the equivalent circuit of X_1. The mapping information is effectively exchanged leading to two possible offspring. If neither parent contains a mapping then this operator falls back to swapping the model types.
- *One-point row crossover*: classic one-point-crossover is performed on the matrix rows. A row r is selected randomly and an offspring is generated by taking rows 1 to $r-1$ from X_1 together with rows r to p from X_2. A second offspring can be generated by reversing the roles of X_1 and X_2. In circuit terms, this means that the outer $p-r$ parallel branches of the parent circuits are exchanged in order to generate two offspring.
- *Exchange a branch*: a branch $b \in \{GS, GD, DS\}$ is chosen randomly and exchanged between both parent circuits with all attached parallel branches.

However, in contrast to the mutation operator, only a single recombination operator can be active at a time. Fig. 9 illustrates the application of the branch exchange operator (only one of the two possible children is shown).

3) Repair function: Finally, the reader may have noted that without further modifications, the algorithm as described so far may generate invalid circuits. Therefore at the end of each genetic operator an extra check on the circuit topology is performed, and the individual is repaired if necessary. For example, if the circuit is not closed, a random branch is added to close the circuit.

E. Fitness Function

1) Decoding a solution matrix into a circuit netlist: The fitness function is the core of the genetic algorithm. It maps an individual from the population onto a scalar score which represents the difference between the model and the transistor device data. In doing so, it must decode the matrix representing the individual into an equivalent circuit model. To make this possible, the fitness function uses an eXtensible Markup Language (XML) [37] file that contains the approximation equations (in a netlist compatible format) for each possible element that can occur in the circuit.

Fig. 10 shows an example of two elements that the XML file may contain. The first element contains the equation of a capacitor (which occurs in the *GS* branch) approximated by

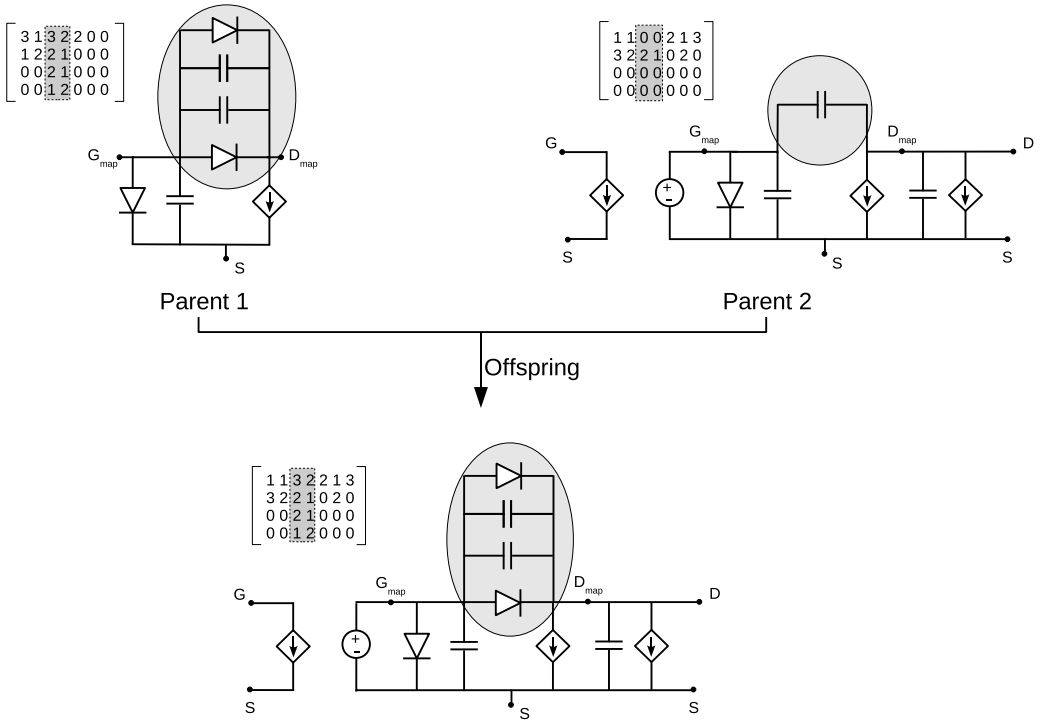


Figure 9: Example of applying the *"Exchange a branch"* crossover operator. In this case the *GD* branch is randomly chosen and exchanged between the two parents (including all parallel branches). Only one of the two possible children is shown in the figure: the child which inherits everything form the parent on the right, except the *GD* branch, which it inherits from the parent on the left. This is equivalent to exchanging the 3^{rd} and 4^{th} columns in the matrix encoding.

the Chalmers model. The second element shown in Fig. 10 holds the equations for a linear difference mapping on the gate terminal. Thus, as is clear from the figure, XML is a markup language (i.e., it annotates existing data) that permits standardized representation of structured data. The choice for XML was natural since it is the defacto standard for structured data representation, and support for it is built-in to virtually every programming language and development environment. It also allows the extension of this approach to new equivalent circuit models and mapping types in the future.

Furthermore, once data information is stored in XML format it becomes very easy to manipulate and extend. In particular querying the XML data for specific information can be easily done. This is made possible through the XML Path Language (XPath) [39]. For example, if one wanted to uniquely retrieve the first element listed in Fig. 10, one would perform the following XPath query:

//**Element**[@**model**='*Chalmers*'] and [@**branch**='*GS*'] and [@**type**='*Capacitor*']

Thus a matrix that represents an encoded circuit based on (11) can be easily mapped onto a series of XPath calls that collect the necessary elements that make-up the circuit. These elements are then combined and merged into a base template netlist in order to arrive at the final netlist that fully implements the circuit. This netlist can then be executed by the circuit simulator.

2) Mapping a circuit to a fitness value: The next step is to produce a scalar fitness value for a given netlist, this is the task of the fitness function. The fitness function should assess the quality of a given netlist against the criteria selected by the designer. In this paper we have restricted ourselves to minimizing the error between the circuit model and the true device data. Though other metrics (such as a stability related term) could be included as well.

Let N be a netlist that implements a particular circuit. N is the result of decoding an encoded matrix X according to the procedure described in subsection II-E1. The accuracy of N as a model of the transistor in question, is calculated based on the training data, such as the *DC* and bias-dependent *S*-parameter data. Let this data be denoted by $\mathbf{y}$ and the number of data samples by n. In addition, the empirical equations in each branch of the equivalent circuit model contain a number of parameters that must be set (e.g., the P_{11} and P_{21} parameters from Fig. 10). Let θ_E denote the values of these parameters and let N_{θ_E} be the circuit represented by netlist N whose circuit parameters have been set to θ_E. Let $\mathbf{g}(\cdot)$ be a function that represents the execution of the netlist by the circuit simulator. $\tilde{\mathbf{y}} = \mathbf{g}(N_{\theta_E})$ then denotes the prediction of the training data by N_{θ_E}. In order to achieve maximum accuracy, θ_E must be optimized to an optimal value θ_E^*. This can be performed by a circuit simulator (in this paper we use Agilent Advanced Design System (ADS) [38]). The optimization routines present in the circuit simulator will optimize θ_E in order to accurately fit the empirical equivalent circuit model to the training data. Denote this optimized circuit model by $N_{\theta_E^*}$ with

$$\theta_E^* = \arg\min_{\theta_E} h(\mathbf{y}, \mathbf{g}(N_{\theta_E})) \tag{16}$$

```
</NetlistElements>

  <Element model="Chalmers" branch="GS" type="Capacitor">
    <SDD>SDD:SDD2P3  Gc1 Sc1 Dc1 Sc1  I[1,1]=Qgs(_v1, _v2)  I[2,0]=0</SDD>
    <Parameters>
      P11 = 0.423083438 opt{ unconst };
      P21 = 0.012810991 opt{ unconst };
      Cgs0 = 6.75906440e-13 opt{ unconst };
    </Parameters>
    <Functions>
      Qgs(v1, v2) = Cgs0*(P11*v1+ln(cosh(P11*v1)))*(1+tanh(P21*v2))/P11
    </Functions>
  </Element>

  ...

  <Mapping terminal="Gate" type="DiffMapping">
    <Parameters>
      ag0 = 0 opt{  +/- 5 };
      ag1 = 0 opt{  +/- 5 };
      ag2 = 0 opt{  +/- 5 };
      bg0 = 0 opt{  +/- 5 };
      bg1 = 0 opt{  +/- 5 };
      bg2 = 0 opt{  +/- 5 };
    </Parameters>

    <Functions>
      Ig(Vg, Vd) = ag0+ag1*Vg+ag2*Vd;
      Qg(Vg, Vd) = bg0+bg1*Vg+bg2*Vd;
    </Functions>
  </Mapping>
  ...
</NetlistElements>
```

Figure 10: Example XML fragment for storing the implementation of each of the elements that may appear in a circuit. The approximation for a circuit element/mapping is split into different parts: the actual equation (the <*Functions*> tag), where the equation is used in the circuit netlist (<*SDD*> tag), and the parameters that occur in the equation (<*Parameters*> tag). The *opt{...}* specifiers indicate to the circuit simulator that will process this netlist that those parameters are optimizable within the given bounds. In this example the circuit simulator used is ADS [38].

The score returned by the function $h(\cdot)$, and minimized by the circuit simulator, is a weighted sum (using weights w_i with $||\mathbf{w}|| = 1$) of the absolute error on the *DC* and the real and imaginary parts of the *S*-parameters, i.e.

$$h(\mathbf{y},\tilde{\mathbf{y}}) = \sum_{i=1}^{9} w_i \cdot ASE(y_{o_i},\tilde{y_{o_i}}) \tag{17}$$

Where y_{o_i} and $\tilde{y_{o_i}}$ represent the true and predicted values of each output, i.e., the drain current I_d, and the real and imaginary *S*-parameters ($o_i,\tilde{o_i} \in \{I_d, real(S_{11}), imag(S_{11}), real(S_{12}), imag(S_{12}), real(S_{21}), imag(S_{21}), real(S_{22}), imag(S_{22})\}$). The Average Scaled Error (ASE) between y_{o_i} and $\tilde{y_{o_i}}$ is defined as

$$ASE(y_{o_i},\tilde{y_{o_i}}) = \frac{1}{n}\sum_{k=1}^{n} \frac{|y_{o_{i_k}} - \tilde{y_{o_{i_k}}}|}{|\max(y_{o_i}) - \min(y_{o_i})|} \tag{18}$$

where i, $i = 1,2,...,9$ denotes the index of outputs, and k, $k = 1,2,...,n$ denotes the index of the data samples.

Note that so far we have only discussed the optimization of the parameters in the equivalent circuit models themselves. If a mapping structure is present in an individual solution, the parameters of that mapping (denoted by θ_M) need to be optimized as well (e.g., the $ag_0,...,bg_0$ parameters from Fig. 10). However, the effective use of mapping structures requires that the circuit without mapping is already of reasonable quality [11]. Therefore, if a solution contains one or more mapping structures, two optimizations are performed. First the circuit is optimized over θ_E without the mapping structure in order to find $N_{\theta_E^*}$ as in (16). Then, for the fixed value of θ_E^*, an optimization is performed over θ_M to find the optimal set of mapping parameters θ_M^*:

$$\theta_M^* = \arg\min_{\theta_M} h(\mathbf{y}, \mathbf{g}(N_{\{\theta_E^*,\theta_M\}})) \tag{19}$$

Let $N_{\{\theta_E^*,\theta_M^*\}}$ be the netlist N whose parameters are set to these optimal values. The final fitness value returned for a given netlist N is thus

$$fitness(N) = h(\mathbf{y}, \mathbf{g}(N_{\{\theta_E^*,\theta_M^*\}})) \tag{20}$$

The full search by the genetic algorithm in order to find the optimal circuit topology with optimal mapping structure N^* can then be written as

$$N^* = \arg\min_{N} fitness(N) \tag{21}$$

This whole process is illustrated graphically in in Fig. 11.

3) Parallel execution of circuit simulations: The repeated optimization of each solution generated by the genetic algorithm can be expected to be computationally expensive. Luckily, a major advantage of evolutionary algorithms is that they naturally allow for parallelization. The parallel computing model is Single-Process-Multiple-Data thus the fitness of each circuit in the population can be calculated independently on different CPU's. Thus the fitness function is implemented so that multiple circuit simulator instances can be run in parallel, each simulating a particular circuit of the population. In addition a fitness cache is used to prevent running the same simulations twice (through selection and elitism the population may contain duplicates), reducing the running time even further. This can be done since the simulations are deterministic.

Finally, note that the circuit topology and type of elements produced by the proposed method are not completely randomly created from scratch, rather they are re-combinations of corresponding branches (i.e., meaningful branches) from various existing models. This helps confine the stability property of the neuro-space mapped models generated from the proposed algorithm to be similar to that of the manually constructed neuro-space mapped models. A possible future study to further ensure stability is to expand the fitness function of the genetic algorithm to include a penalty term related to stability of the model.

III. Examples

A. Evolutionary Modeling of a GaAs MESFET Device

This example illustrates the proposed technique on a large-signal Field-Effect-Transistor (FET) model trained with both DC and bias-dependent S-parameter data. The fine device data is generated using an ADS internal GaAs FET model [13], [25]. The data is available at 20 frequencies (1-20 GHz) and 125 biases ($V_g \in [-1,0]$ V, $V_d \in [0.2,5]$ V).

The equivalent circuit models used as building blocks for the genetic algorithm are the Curtice model [24], the Materka

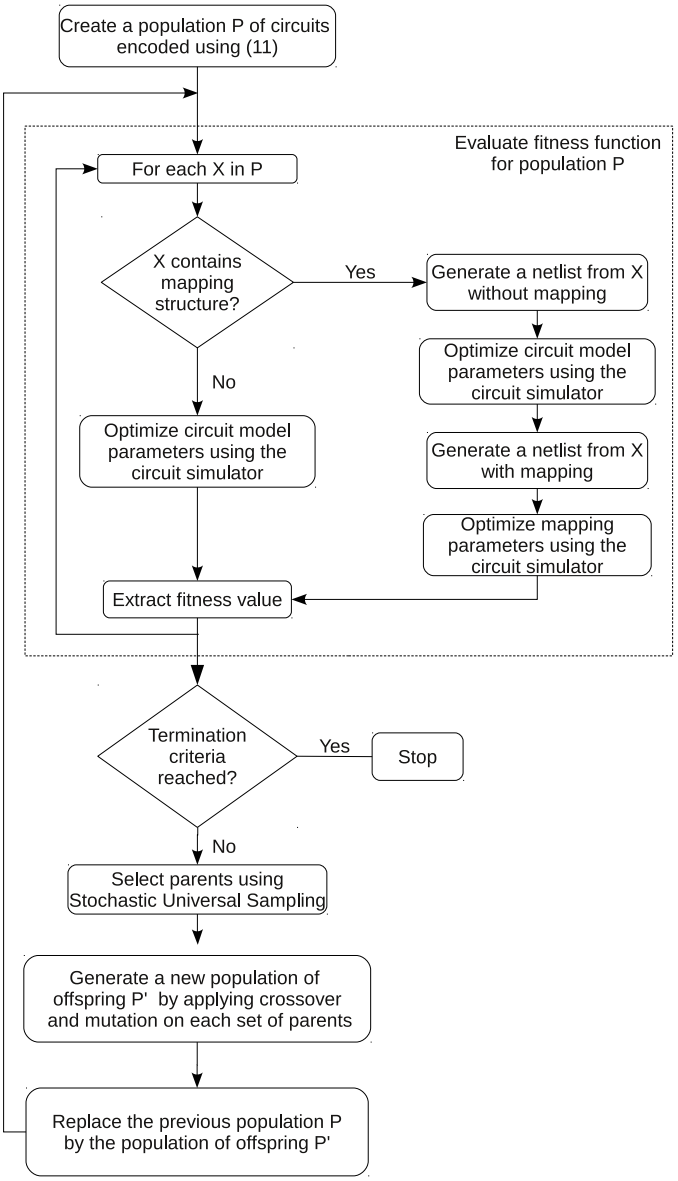
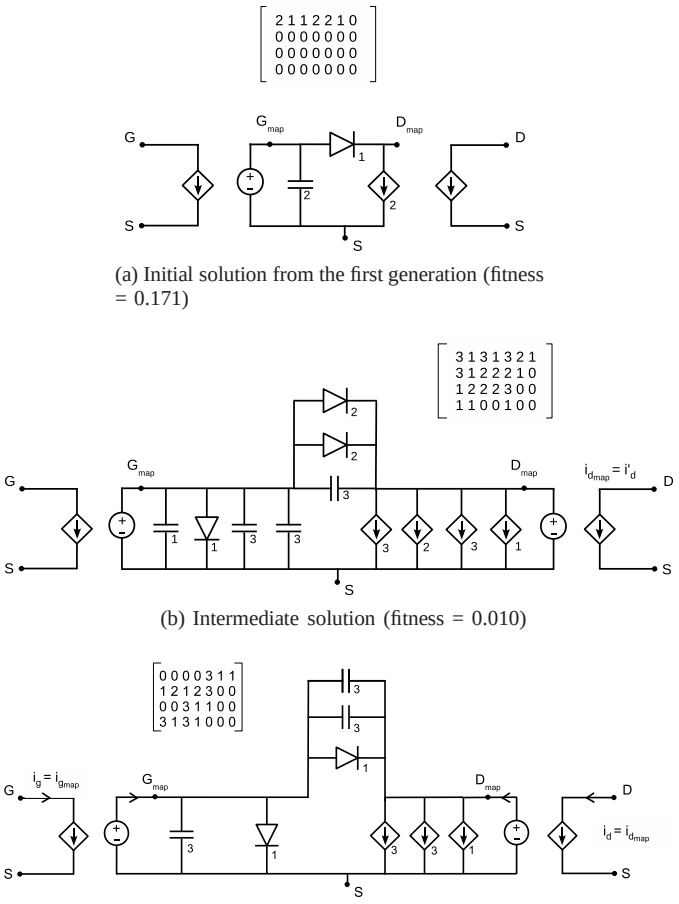


Figure 11: Flowchart for the evolution of optimal Neuro-SM models.



(a) Initial solution from the first generation (fitness = 0.171)

(b) Intermediate solution (fitness = 0.010)

(c) Final solution after 100 generations (fitness = 0.001)

Figure 12: Genetic Algorithm generated solutions for the MESFET example.

model [26], and the Chalmers model [27]. The mapping structures included are input mapping, output mapping, and difference mapping. For this paper we build upon the neuro-space mapping approach described in [13] and [14]. However, now the focus is on type selection of the mapping structure while at the same time achieving best possible quality of the equivalent circuit. This is made possible by mixing model types and performing circuit topology optimization. We wish to keep the mapping structure simple to make the model more efficient and robust. For this reason we use a linear ANN for f_{IM} and f_{OM} which is a simpler version of the general nonlinear ANN-based mapping used in [13].

The maximum number of parallel branches (p) was set to 4. If we then calculate the number of possible circuits (i.e., the size of the search space) according to (12) we see that this is more than 9.50×10^7. Far too large to explore manually.

All tests were run on a Matlab 7.8 R2009a platform [40] utilizing ADS 2006 [38], the Matlab Genetic Algorithm & Direct Search, and parallel computing toolboxes. Simulations were run on a Quad Core Intel machine with 2GB main memory running Ubuntu Linux 8.10. Using the local scheduler from the parallel computing toolbox, this allowed for running 4 ADS simulations in parallel. If also the distributed scheduler is available, the implementation can also seamlessly run on a large cluster or grid of machines, without requiring any modifications to the code.

During the execution of the genetic algorithm, ADS was configured to perform 10 optimization iterations using the standard gradient-based method per circuit fitness evaluation. The time for one fitness evaluation is roughly 180 seconds on average for an individual with no mapping and double that if one or more mapping structures are present. The initial values for each of model parameters were set to good, sensible values. The final solution was optimized for an additional 200 iterations. If ADS failed to converge during the coarse model parameter optimization, a score of 1000 was assigned to the individual causing the failure. If the failure occurred during the mapping optimization, the score of the individual without mapping is returned. The population size was set to 20, the maximum number of generations to 100, the crossover

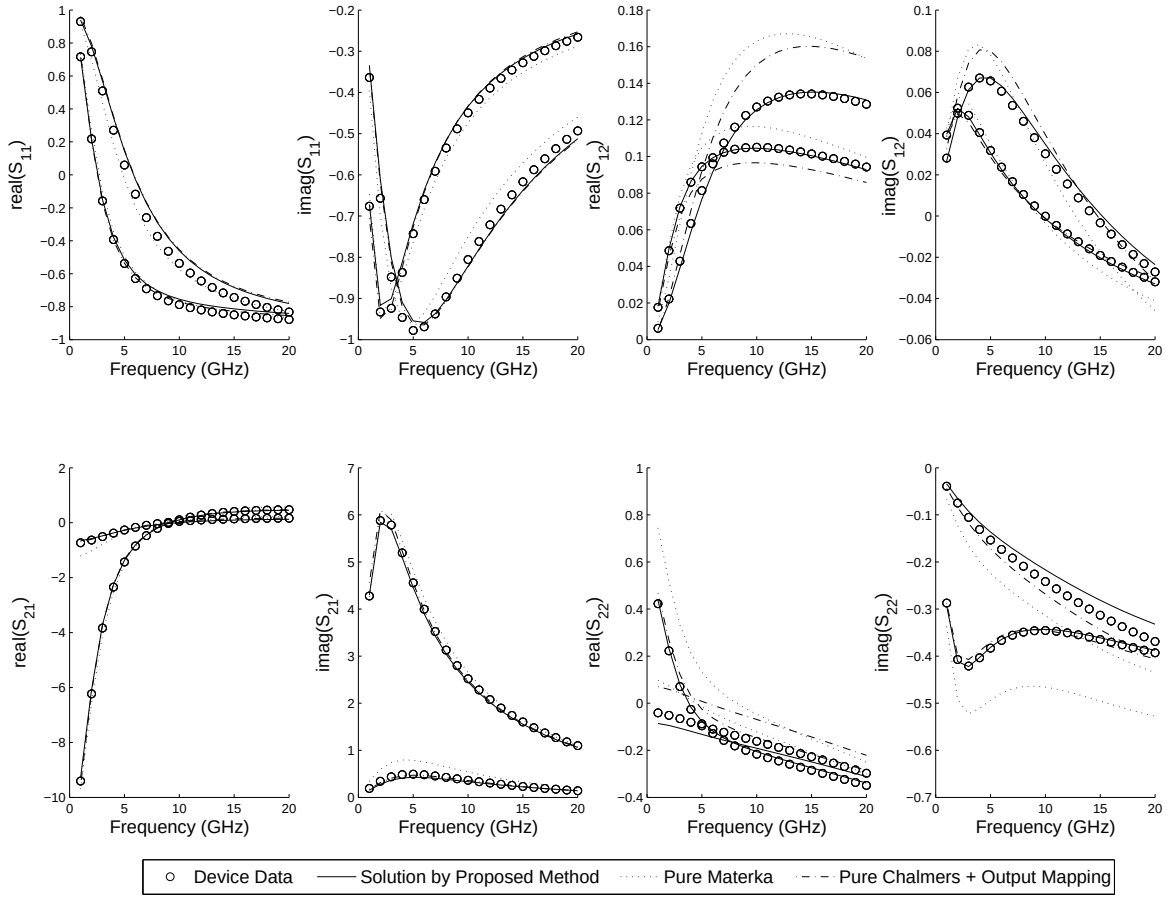


Figure 13: *S*-parameter plot for $(V_g, V_d) \in \{(-1,0),(0.2,5)\}$ for the MESFET example.

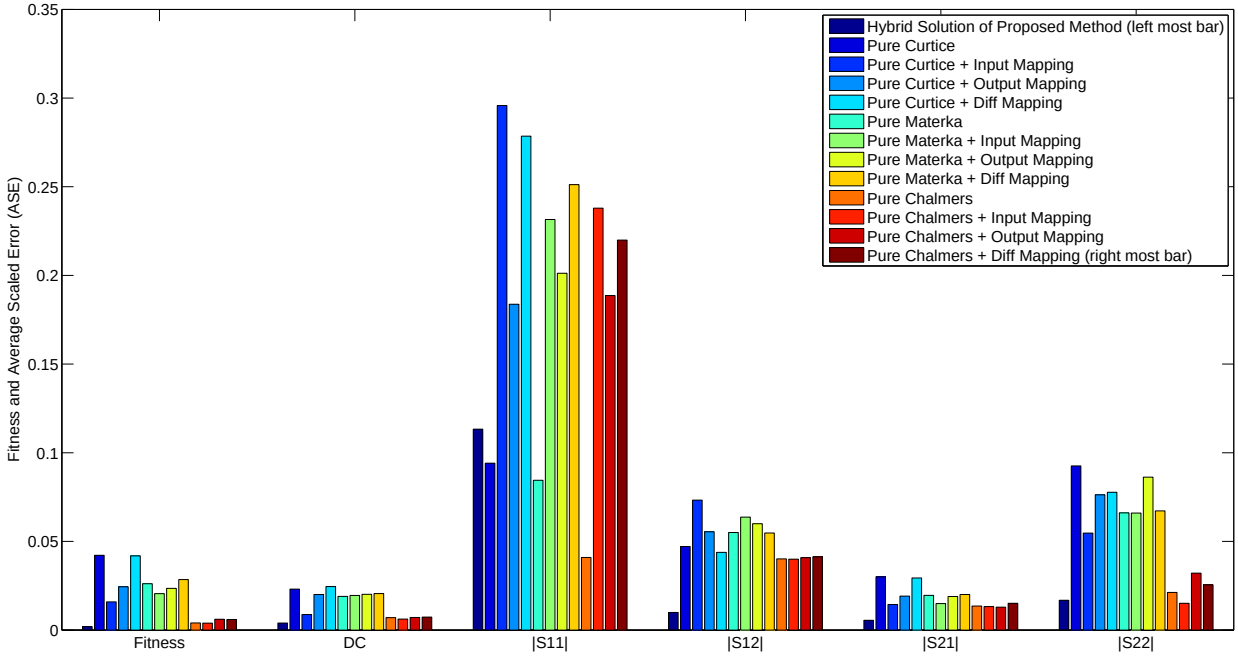


Figure 14: Comparison of the model from the proposed evolutionary modeling method with pure models on the MESFET modeling problem (Average Scaled Error). A lower fitness means that the overall error of *DC* and *S*-parameters is lower and that the model is more accurate. The solution from the proposed method is a hybrid model with mappings and gives the best accuracy overall.

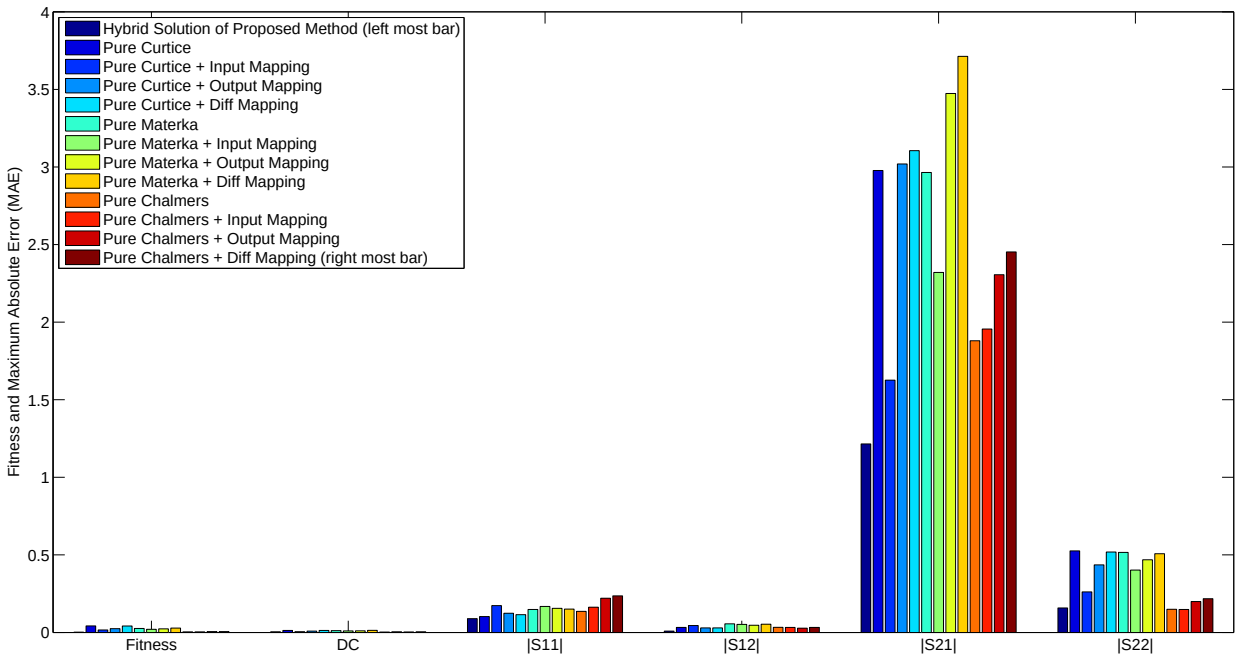


Figure 15: Comparison of the model from the proposed evolutionary modeling method with pure models on the MESFET modeling problem (Maximum Absolute Error). A lower fitness value means a more accurate model. The solution from the proposed method, which is a hybrid model with mappings, gives the best accuracy.

Table II: Comparison of the model from the proposed evolutionary modeling method with pure models on the MESFET modeling problem. A lower fitness values means a more accurate model. The solution from the proposed method is a hybrid model with mappings and gives the best accuracy overall.

		Average Scaled Error (ASE)				
	Fitness	**DC**	**|S11|**	**|S12|**	**|S21|**	**|S22|**
Solution of Proposed Method	**0.0019**	**0.0040**	0.1133	**0.0099**	**0.0055**	0.0168
Pure Curtice	0.0422	0.0231	0.0941	0.0471	0.0300	0.0926
Pure Curtice + Input Mapping	0.0159	0.0087	0.2958	0.0733	0.0144	0.0547
Pure Curtice + Output Mapping	0.0244	0.0200	0.1838	0.0554	0.0191	0.0763
Pure Curtice + Diff. Mapping	0.0419	0.0245	0.2785	0.0438	0.0294	0.0777
Pure Materka	0.0261	0.0190	0.0845	0.0550	0.0196	0.0661
Pure Materka + Input Mapping	0.0205	0.0195	0.2315	0.0637	0.0149	0.0660
Pure Materka + Output Mapping	0.0235	0.0202	0.2012	0.0600	0.0189	0.0863
Pure Materka + Diff. Mapping	0.0285	0.0206	0.2512	0.0548	0.0200	0.0672
Pure Chalmers	0.0040	0.0070	**0.0410**	0.0401	0.0136	0.0212
Pure Chalmers + Input Mapping	0.0039	0.0062	0.2379	0.0400	0.0133	**0.0151**
Pure Chalmers + Output Mapping	0.0061	0.0071	0.1886	0.0408	0.0129	0.0321
Pure Chalmers + Diff. Mapping	0.0059	0.0073	0.2199	0.0413	0.0151	0.0255

		Maximum Absolute Error (MAE)				
	Fitness	**DC**	**|S11|**	**|S12|**	**|S21|**	**|S22|**
Solution of Proposed Method	**0.0019**	0.0043	**0.0891**	**0.0087**	**1.2154**	0.1583
Pure Curtice	0.0422	0.0132	0.1028	0.0323	2.9773	0.5257
Pure Curtice + Input Mapping	0.0159	0.0055	0.1733	0.0448	1.6263	0.2616
Pure Curtice + Output Mapping	0.0244	0.0090	0.1241	0.0294	3.0194	0.4360
Pure Curtice + Diff. Mapping	0.0419	0.0132	0.1147	0.0298	3.1051	0.5182
Pure Materka	0.0261	0.0116	0.1483	0.0557	2.9645	0.5163
Pure Materka + Input Mapping	0.0205	0.0100	0.1678	0.0519	2.3200	0.4023
Pure Materka + Output Mapping	0.0235	0.0105	0.1555	0.0468	3.4738	0.4683
Pure Materka + Diff. Mapping	0.0285	0.0139	0.1510	0.0533	3.7128	0.5077
Pure Chalmers	0.0040	**0.0027**	0.1362	0.0332	1.8802	0.1494
Pure Chalmers + Input Mapping	0.0039	0.0044	0.1631	0.0327	1.9553	**0.1480**
Pure Chalmers + Output Mapping	0.0061	0.0034	0.2208	0.0275	2.3055	0.1995
Pure Chalmers + Diff. Mapping	0.0059	0.0044	0.2362	0.0324	2.4523	0.2176

probability to 0.8, and 3 elite individuals were used. The initial population consists of a pure equivalent circuit model of each type (Curtice, Chalmers, Materka) with $p = 2$ and randomly generated circuits, also with $p = 2$. The selection function utilized is Stochastic Universal Sampling [41]. Once the evolution has terminated, we record the Average Scaled Error and Maximum Absolute Error (MAE) for the best solution found. Both error metrics are calculated between the device data and the predicted model response over the *DC* and *S*-parameter responses. The MAE between the true values $\mathbf{y}$ and the predicted values $\tilde{\mathbf{y}}$ is defined as

$$MAE(\mathbf{y},\tilde{\mathbf{y}}) = \max(|\mathbf{y}-\tilde{\mathbf{y}}|) \tag{22}$$

For the MESFET problem, the final solution and two intermediate solutions generated by the genetic algorithm are shown in Fig. 12. The best solution achieved by the genetic algorithm is a hybrid model that mixes the Curtice and Chalmers models with suitable mapping structures:

- Mapping: Input mapping on the gate terminal and drain terminal
- *GS* branch: Curtice Diode, Chalmers Capacitor
- *GD* branch: Curtice Diode, Chalmers Capacitor, Chalmers Capacitor
- *DS* branch: Chalmers Source, Chalmers Source, Curtice Source

Note that we have duplicate model elements in the *GD* and *DS* branches. Although these elements have the same type, their parameters are different (due to the optimization) thus their behavior is not the same.

For purposes of comparison, the data was also modeled using pure forms of each of the different equivalent circuit model types. We use the term "pure model" to represent conventional Neuro-SM models where all branches of the equivalent circuit belong to the same model type (i.e., no mix of branches from different types of models), and there is only one type of mapping for both the gate and drain terminals (e.g., the standard Chalmers model with input mapping on both gate and drain terminals). Fig. 13 shows a plot of the *S*-parameters at 2 specific bias values for the genetic algorithm solution and two other models. From the figure it can be observed that a pure Materka model with input mapping performs better than a pure Curtice model. A pure Chalmers model with output mapping performs even better.

Fig. 14 shows how the evolved solution compares to each of the pure models (the exact numbers can be found in Table II). As can be seen from Fig. 14, the circuit generated by the genetic algorithm is the best solution (lowest fitness) over all other types of models under test. Regarding the average error it performs best on *DC*, S_{12}, S_{21} and gives the same performance as the best pure model on S_{22}. The lesser performance on S_{11} turns out to be due to a lower accuracy on the real part of S_{11} that gets magnified when calculating the Average Scaled Error of the magnitude (the generated solution is the best solution of the imaginary component). Fig. 14 shows the accuracy in terms of average error. Worst case performance is shown in Fig. 15 and the general tendency is the same. The solution found by the proposed evolutionary Neuro-SM algorithm performs equal to (for *DC* and S_{22}) or better than (for S_{11}, S_{12} and S_{21}) any of the pure models. Especially for S_{21} the worst case accuracy of the proposed solution is significantly better.

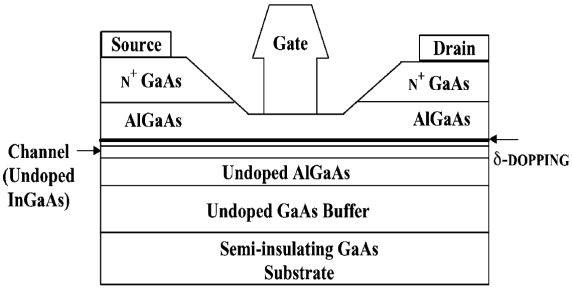


Figure 16: Physical structure of a HEMT device used for generating fine data in the MINIMOS physics based device simulator.

Note that in this example, the data generator is an existing ADS model for convenience of illustration. Note also that the data generator is chosen such that the equations producing the data clearly differ from that of the coarse models used for mapping (Curtice, Chalmers, Materka). Thus this example confirms that, eventhough the given coarse models used for mapping are different from that of the data generator, the proposed method can map the coarse model onto the data with good accuracy. This confirmation provides evidence that in reality when the fine model is unknown, the proposed method will produce an accurate model by mapping existing models closely to the data.

B. Evolutionary Modeling of a HEMT Device

The High Electron Mobility Transistor (HEMT) device is important in high frequency circuit design. In this example, the proposed technique is used to learn physics-based data of a HEMT device [13], with the training data (DC and bias dependent S-parameter data) generated from a physics-based device simulator, MINIMOS [42], by solving the device Poisson equations. The data is available at 40 frequencies (1-40GHz) and 125 biases ($V_g \in [-5,-1]$ V, $V_d \in [0,3]$ V).

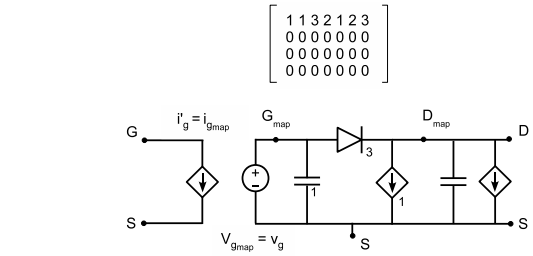
The HEMT structure used in setting up the physics-based simulator is shown in Fig. 16 [13]. Since this is a more complex modeling problem we also add the necessary extrinsic components, though they remain fixed throughout the evolution. The same configuration (chomosomal encoding, genetic algorithm settings, ADS settings) as the first example is used.

The final solution and two intermediate solutions found by the genetic algorithm are depicted in Fig. 17. The final solution is a hybrid model with mapping and is made up of the following elements:

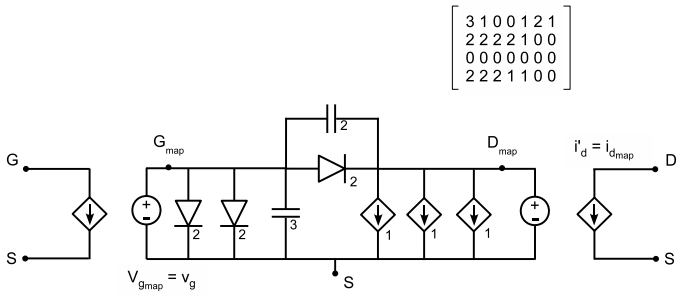
- Mapping: Input mapping on the gate, input mapping and output mapping on the drain
- *GS* branch: Chalmers Capacitor, Curtice Capacitor
- *GD* branch: Materka Diode, Curtice Diode
- *DS* branch: Materka Source, Materka Source

The *S*-parameter plot for this problem is shown in Fig. 18. The genetic algorithm generated solution shows clear improvement over the other pure equivalent circuit models. Figs. 19 and

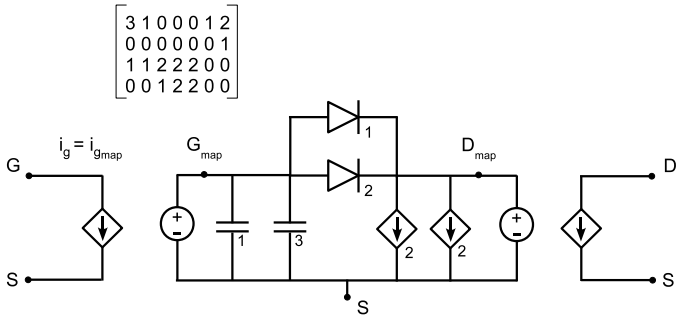
20 show how the accuracy compares with each of the pure models and models with pre-determined mapping structures. The exact numbers can be found in Table III. Again we see that the genetic algorithm solution compares favorably to the other pure models, both in worst case error and in average error. From Fig. 19 it is interesting to note that even though pure Curtice and pure Materka models have high errors on S_{11}, the solution generated by the genetic algorithm performs much better while consisting mostly of Materka and Curtice elements.



(a) Initial solution from the first generation (fitness = 1.070)



(b) Intermediate solution (fitness = 0.256)



(c) Final solution after 100 generations (fitness = 0.109)

Figure 17: Genetic algorithm generated solutions for the HEMT example.

This is a clear example of how mixing model types can improve accuracy. For the other components the evolved solution is competitive with a Chalmers-based model, with the former having an overall advantage (lower fitness) due to the improved worst case behavior.

IV. Conclusion and Future Work

We address the problem when existing empirical models have difficulty fitting new devices well. Previous research in this direction has shown that the Neuro-SM methods that augment equivalent circuit models with mapping structures can deliver promising results. However, determining the optimal combination of mapping structure and equivalent circuit model remains a user intensive process.

In this paper we have presented an evolutionary approach to Neuro-SM based modeling of microwave devices that tackles this. Good results were demonstrated on two modeling problems. Through the use of a genetic algorithm, the search for the optimal hybrid combination can be performed more efficiently allowing for potentially large gains in accuracy and performance.

A disadvantage of being based on evolutionary algorithms is that results are not deterministic. However, a strength of this approach is that, since the initial population of the genetic algorithm can be seeded with the existing models, for a particular new device the solution produced by the evolutionary process can be guaranteed to be at least as good as what is currently available. Though in the majority of cases, the generated solution will be superior. A second advantage of evolutionary search is that it runs fully autonomously. Its applicability is limited only by the available computing power (which has currently been commoditized due to the rise of multi-core CPUs, clusters, grids and clouds). Our algorithm has taken advantage of this feature and thus naturally scales with the increasing computing power that is made available.

Acknowledgments

This work was supported in part by a grant from the Fund for Scientific Research (FWO), Flanders and in part by the Natural Sciences and Engineering Research council of Canada.

References

[1] Q. J. Zhang, K. Gupta, and V. Devabhaktuni, "Artificial neural networks for RF and microwave design: from theory to practice," *IEEE Transactions on Microwave Theory and Techniques*, vol. 51, pp. 1339–1350, Apr. 2003.

[2] J. E. Rayas-Sanchez, "Em-based optimization of microwave circuits using artificial neural networks: The state-of-the-art," *IEEE Transactions on Microwave Theory and Techniques*, vol. 52, pp. 420–435, Jan. 2004.

[3] P. Burrascano, S. Fiori, and M. Mongiardo, "A review of artificial neural networks applications in microwave computer-aided design," *International Journal of RF and Microwave Computer Aided Engineering*, vol. 9, pp. 158–174, May 1999.

[4] V. Rizzoli, A. Neri, D. Masotti, and A. Lipparini, "A new family of neural network-based bidirectional and dispersive behavioural models for nonlinear rf/microwave subsystems," *International Journal of RF and Microwave Computer Aided Engineering*, vol. 12, pp. 51–70, Jan 2002.

[5] Q. J. Zhang and K. C. Gupta, *Neural Networks for RF and Microwave Design (Book + Neuromodeler Disk)*. Norwood, MA, USA: Artech House, Inc., 2000.

[6] A. H. Zaabab, Q. J. Zhang, and M. Nakhla, "A neural network modeling approach to circuit optimization and statistical design," *IEEE Transactions on Microwave Theory and Techniques*, vol. 43, pp. 1349–1358, Jun. 1995.

[7] 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*, vol. 18, pp. 485–494, Jun. 2009.

[8] J. Bandler, Q. Cheng, S. Dakroury, A. Mohamed, M. Bakr, K. Madsen, and J. S/ondergaard, "Space mapping: the state of the art," *IEEE Transactions on Microwave Theory and Techniques*, vol. 52, pp. 337–361, Jan. 2004.

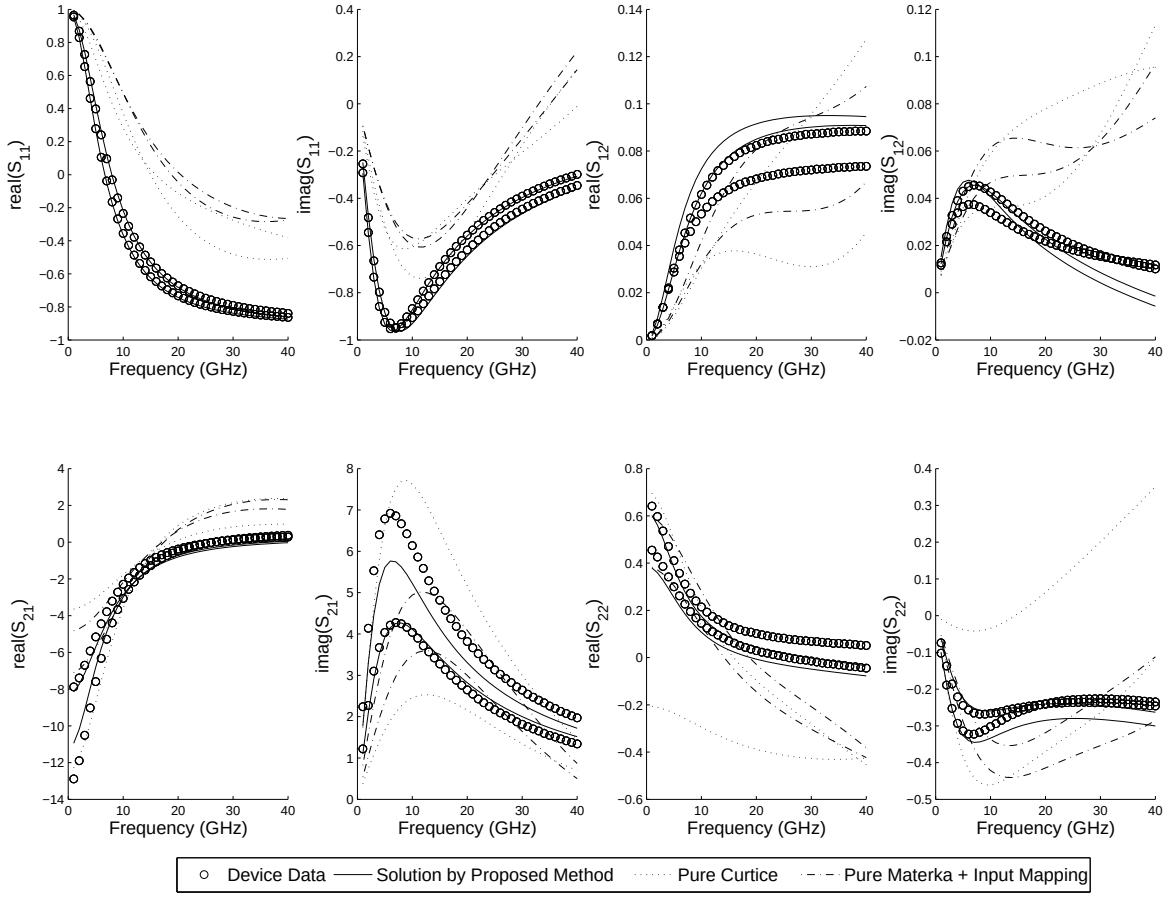


Figure 18: S-parameter plot for $(V_g, V_d) \in \{(-0.4, 0.1), (-0.2, 3)\}$ for the HEMT example.

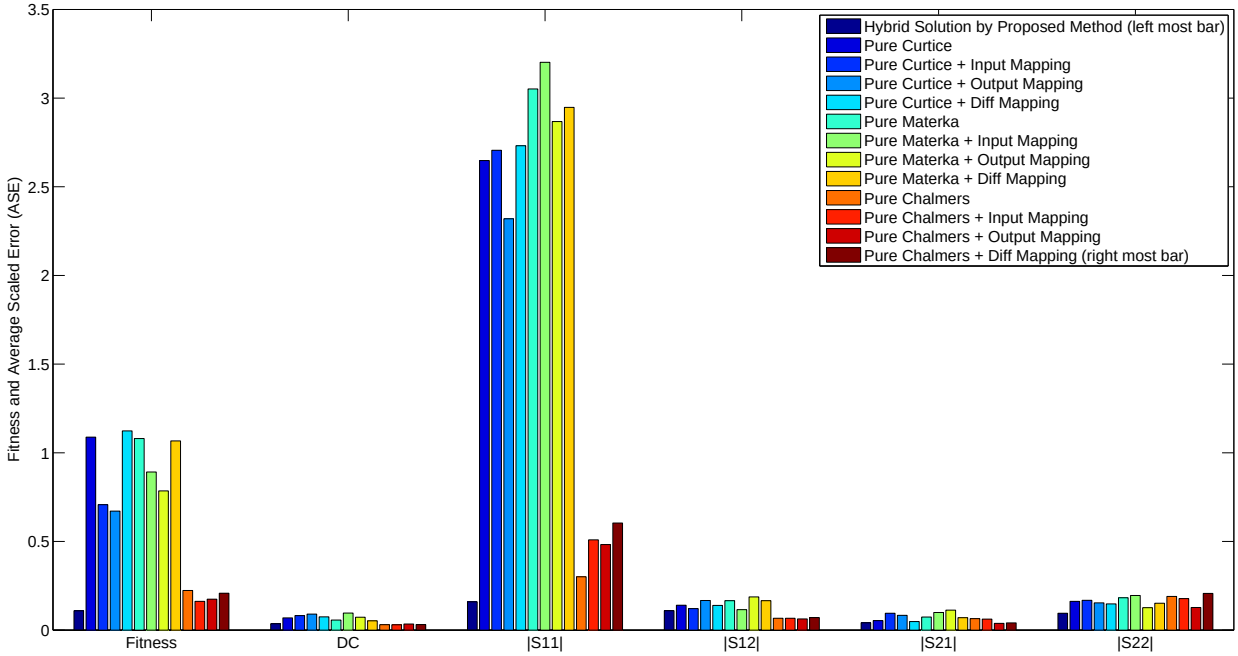


Figure 19: Comparison of the model from the proposed evolutionary modeling method with pure models on the HEMT modeling problem (Average Scaled Error). A lower fitness value means a more accurate model. The solution from the proposed method is a hybrid model with mappings and gives the best accuracy overall.

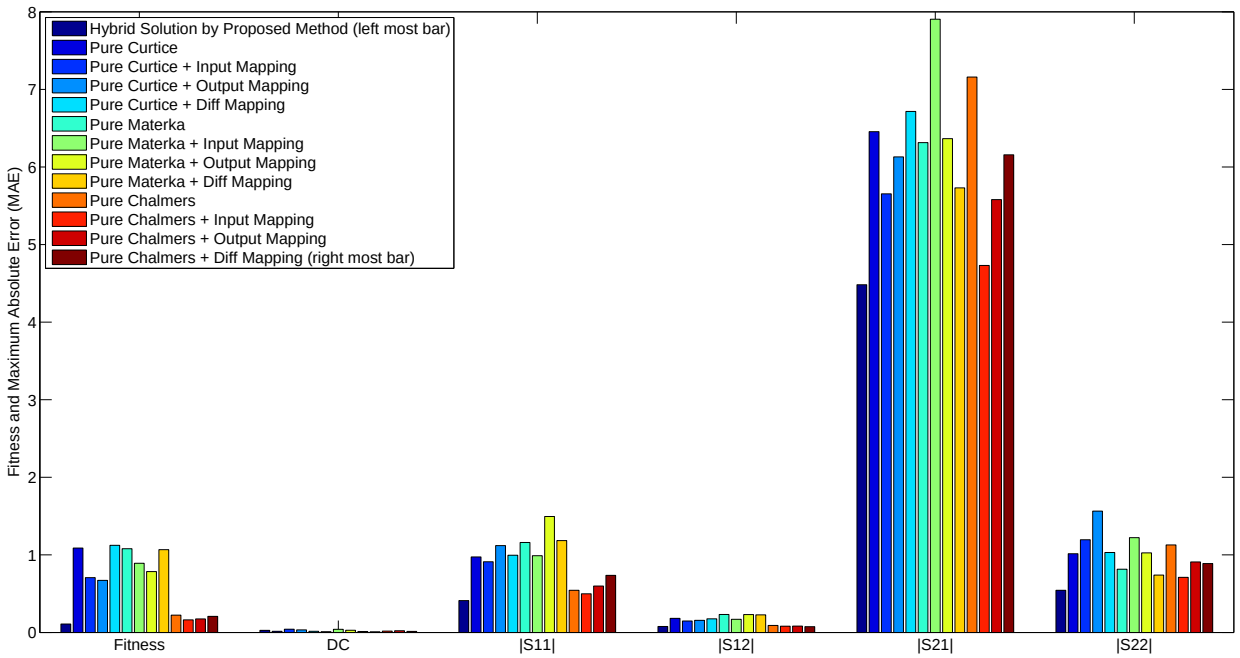


Figure 20: Comparison of the model from the proposed evolutionary modeling method with pure models on the HEMT modeling problem (Maximum Absolute Error). A lower fitness value means a more accurate model. The solution from the proposed method, which is a hybrid model with mappings, gives the best accuracy.

Table III: Comparison of the model from the proposed evolutionary modeling method with pure models on the HEMT modeling problem. A lower fitness values means a more accurate model. The solution from the proposed method is a hybrid model with mappings and gives the best accuracy overall.

		Average Scaled Error (ASE)				
	Fitness	**DC**	**|S11|**	**|S12|**	**|S21|**	**|S22|**
Solution of Proposed Method	**0.1091**	0.0361	**0.1598**	0.1096	0.0411	**0.0953**
Pure Curtice	1.0881	0.0687	2.6477	0.1404	0.0537	0.1623
Pure Curtice + Input Mapping	0.7072	0.0817	2.7057	0.1213	0.0949	0.1680
Pure Curtice + Output Mapping	0.6713	0.0902	2.3196	0.1671	0.0833	0.1536
Pure Curtice + Diff. Mapping	1.1234	0.0751	2.7313	0.1392	0.0483	0.1477
Pure Materka	1.0802	0.0567	3.0519	0.1661	0.0739	0.1826
Pure Materka + Input Mapping	0.8916	0.0964	3.2018	0.1152	0.0996	0.1954
Pure Materka + Output Mapping	0.7852	0.0725	2.8681	0.1874	0.1126	0.1267
Pure Materka + Diff. Mapping	1.0669	0.0526	2.9479	0.1657	0.0703	0.1513
Pure Chalmers	0.2235	0.0307	0.3010	0.0670	0.0645	0.1901
Pure Chalmers + Input Mapping	0.1625	**0.0301**	0.5088	0.0669	0.0619	0.1777
Pure Chalmers + Output Mapping	0.1745	0.0341	0.4826	**0.0633**	**0.0384**	0.1278
Pure Chalmers + Diff. Mapping	0.2080	0.0312	0.6037	0.0708	0.0406	0.2072

		Maximum Absolute Error (MAE)				
	Fitness	**DC**	**|S11|**	**|S12|**	**|S21|**	**|S22|**
Solution of Proposed Method	**0.1091**	0.0268	**0.4114**	0.0773	**4.4820**	**0.5441**
Pure Curtice	1.0881	0.0149	0.9737	0.1822	6.4553	1.0153
Pure Curtice + Input Mapping	0.7072	0.0423	0.9116	0.1483	5.6536	1.1947
Pure Curtice + Output Mapping	0.6713	0.0337	1.1196	0.1568	6.1296	1.5647
Pure Curtice + Diff. Mapping	1.1234	0.0149	0.9951	0.1763	6.7160	1.0309
Pure Materka	1.0802	0.0105	1.1604	0.2323	6.3137	0.8150
Pure Materka + Input Mapping	0.8916	0.0412	0.9891	0.1696	7.9045	1.2219
Pure Materka + Output Mapping	0.7852	0.0282	1.4954	0.2308	6.3656	1.0266
Pure Materka + Diff. Mapping	1.0669	0.0123	1.1845	0.2276	5.7310	0.7399
Pure Chalmers	0.2235	**0.0089**	0.5436	0.0907	7.1590	1.1284
Pure Chalmers + Input Mapping	0.1625	0.0182	0.4981	0.0816	4.7307	0.7111
Pure Chalmers + Output Mapping	0.1745	0.0226	0.5986	0.0817	5.5792	0.9093
Pure Chalmers + Diff. Mapping	0.2080	0.0145	0.7365	**0.0748**	6.1568	0.8880

[9] J. Bandler, N. Georgieva, M. Ismail, J. Rayas-SÃ¡nchez, and J. . Q. J. Zhang vol. 49, pp. 67-79, "A generalized space mapping tableau approach to device modeling," *IEEE Transactions on Microwave Theory and Techniques*, vol. 49, pp. 67–79, Jan. 2001.
[10] J. Bandler, R. M. Biernacki, S. H. Chen, P. A. Grobelny, and R. H. Hemmers, "Space mapping technique for electromagnetic optimization," *IEEE Transactions on Microwave Theory and Techniques*, vol. 42, pp. 2536–2544, Aug. 1994.
[11] S. Koziel and J. Bandler, "Coarse and surrogate model assessment for engineering design optimization with space mapping," in *Proc. IEEE/MTT-S International Microwave Symposium, Honolulu, HI*, pp. 107–110, Jun. 2007.
[12] S. Koziel and J. Bandler, "Space mapping with multiple coarse models for optimization of microwave components," *IEEE Microwave and Wireless Components Letters*, vol. 18, pp. 1–3, Jan. 2008.
[13] L. Zhang, J. Xu, M. C. E. Yagoub, R. Ding, and Q. J. Zhang, "Efficient analytical formulation and sensitivity analysis of neuro-space mapping for nonlinear microwave device modeling," *IEEE Transactions on Microwave Theory and Techniques*, vol. 53, pp. 2752–2767, Sep. 2005.
[14] L. Zhang, Q. J. Zhang, and J. Wood, "Statistical neuro-space mapping technique for large-signal modeling of nonlinear devices," *IEEE Transactions on Microwave Theory and Techniques*, vol. 56, pp. 2453–2467, Nov. 2008.
[15] G. Doménech-Asensi, J. Hinojosa, J. Martínez-Alajarín, and J. Garrigós-Guerrero, "Empirical model generation techniques for planar microwave components using electromagnetic linear regression models," *IEEE Transactions on Microwave Theory and Techniques*, vol. 53, pp. 3305–3311, Nov. 2005.
[16] E. B. Rahouyi, J. Hinojosa, and J. Garrigós, "Neuro-fuzzy modeling techniques for microwave components," *IEEE Microwave and Wireless Components Letters*, vol. 16, pp. 72–74, Feb. 2006.
[17] S. Koziel, J. W. Bandler, and K. Madsen, "Theoretical justification of space-mapping modeling utilizing a database and on demand parameter extraction," *IEEE Transactions on Microwave Theory and Techniques*, vol. 54, pp. 4316–4322, Dec. 2006.
[18] J. E. Rayas-Sánchez and V. Gutiérrez-Ayala, "EM-based Monte Carlo analysis and yield prediction of microwave circuits using linear-input neural-output space mapping," *IEEE Transactions on Microwave Theory and Techniques*, vol. 54, pp. 4528–4537, Dec. 2006.
[19] M. Steer, J. Bandler, and C. Snowden, "Computer-aided design of RF and microwave circuits and systems," *IEEE Transactions on Microwave Theory and Techniques*, vol. 50, pp. 996–1005, Mar. 2002.
[20] C. M. Snowden, *Semiconductor Device Modeling*. Peter Peregrinus Ltd., London, UK, 1988.
[21] M. A. Khatibzadeh and R. J. Trew, "A large-signal analytical model for the GaAs MESFET," *IEEE Transactions on Microwave Theory and Techniques*, vol. 36, pp. 231–238, Feb. 1988.
[22] K. Lehovec and R. Zuleeg, "Voltagecurrent characteristics of GaAs JFETs in the hot electron range," *Solid State Electron*, vol. 13, pp. 1415–1426, 1970.
[23] C. G. Morton, J. S. Atherton, C. M. Snowden, R. D. Pollard, and M. J. Howes, "A large-signal physical HEMT model," in *Proc. IEEE MTT-S International Microwave Symposium Digest, San Francisco, CA*, vol. 3, pp. 1759–1762, Jun. 1996.
[24] W. R. Curtice, "GaAs MESFET modeling and nonlinear CAD," *IEEE Transactions on Microwave Theory and Techniques*, vol. 36, pp. 220–230, Feb. 1988.
[25] H. Statz, P. Newman, I. W. Smith, R. A. Pucel, and H. A. Haus, "GaAs FET device and circuit simulation in SPICE," *IEEE Transactions on Electron Devices*, vol. 34, pp. 160–169, Feb. 1987.
[26] A. Materka and T. Kacprzak, "Computer calculation of large-signal GaAs FET amplifier characteristics," *IEEE Transactions on Microwave Theory and Techniques*, vol. 33, pp. 129–135, Feb. 1985.
[27] I. Angelov, H. Zirath, and N. Rosman, "A new empirical nonlinear model for HEMT and MESFET devices," *IEEE Transactions on Microwave Theory and Techniques*, vol. 40, pp. 2258–2266, Dec. 1992.
[28] V. I. Cojocaru and T. J. Brazil, "A scalable general-purpose model for microwave FETs including DC/AC dispersion effects," *IEEE Transactions on Microwave Theory and Techniques*, vol. 45, pp. 2248–2255, Dec. 1997.
[29] C. Snowden, "Nonlinear modelling of power FETs and HBTs," *International Journal of Microwave and Millimeter-wave Computer-Aided Engineering*, vol. 6, pp. 219–233, Dec. 1996.
[30] D. E. Root, S. Fan, and J. Meyer, "Technology independent large-signal non quasistatic FET models by direct construction from automatically characterized device data," in *Proc. IEEE 21st European Microwave Conference, Stuttgart, Germany*, pp. 927–932, Oct. 1991.
[31] L. Zhang, J. Xu, M. Yagoub, R. Ding, and Q. Zhang, "Neuro-space mapping technique for nonlinear device modeling and large-signal simulation," in *Proc. IEEE MTT-S International Microwave Symposium Digest, Philadelphia, PA*, pp. 173–176, Jun. 2003.
[32] P. Watson, K. Gupta, and R. Mahajan, "Application of knowledge-based artificial neural network modeling to microwave components," *International Journal of RF and Microwave CAE*, vol. 9, pp. 254–260, May 1999.
[33] L. Zhang, R. Ding, and Q. J. Zhang., "Generalized knowledge-based neural network for microwave modeling," in *9th International Symposium On Antenna Technology and Applied Electromagnetics, Montreal, QC*, pp. 558–561, Aug. 2002.
[34] S. Koziel and J. Bandler, "Support-vector-regression-based output space-mapping for microwave device modeling," in *IEEE MTT-S International Microwave Symposium Digest, Atlanta, GA*, pp. 615–618, Jun. 2008.
[35] H. Ninomiya, S. Wan, H. Kabir, X. Zhang, and Q. Zhang, "Robust training of microwave neural network models using combined global/local optimization techniques," in *Proc. IEEE MTT-S International Microwave Symposium Digest, Atlanta, GA*, pp. 995–998, Jun. 2008.
[36] D. Schreurs, M. Verspecht, S. Vandenberghe, and E. Vandamme, "Straightforward and accurate nonlinear device model parameter-estimation method based on vectorial large-signal measurements," *IEEE Transactions on Microwave Theory and Techniques*, vol. 50, pp. 2315–2319, Oct. 2002.
[37] E. R. Harold and W. S. Means, *XML in a Nutshell*. O'Reilly Media, Inc., Cambridge, Massachusetts, 2004.
[38] Advanced Design System (ADS) 2006, Agilent Technologies, Palo Alto, CA.
[39] P. Walmsley, *XQuery*. O'Reilly Media, Inc., Cambridge, Massachusetts, 2007.
[40] The MathWorks Inc., Natick, MA, USA.
[41] Z. Michalewicz, *Genetic algorithms + data structures = evolution programs (3rd ed.)*. London, UK: Springer-Verlag, 1996.
[42] MINIMOS-NT. Release 2.0, Inst. for Microelectron., Tech. Univ., Vienna, Austria, 2003.

Dirk Gorissen (S'09) received a M.Sc. degree in Computer Science from the University of Antwerp in 2004 and a Masters degree in Artificial Intelligence from the Katholieke Universiteit Leuven in 2007. He continued on to the PhD level at the University of Antwerp and later at Gent University, Belgium where he obtained his PhD in Engineering Science in May 2010. During this time he also worked in research labs in Atlanta, USA and Ottawa, Canada, and he was a member of IBBT, an internationally recognized multidisciplinary ICT research center. Starting February 2010 he joined the Computational Engineering and Design Group at School of Engineering Sciences of Southampton University, UK. His research interests lie in the domain of computational engineering. Particular topics of interest include: global and local surrogate modeling for engineering design exploration and optimization, High Performance Computing, evolutionary computing, machine learning, and software engineering.

Lei Zhang (S'07-M'09) received a B.Eng. degree in Electrical Engineering (major) and Economics (minor) from Tianjin University, Tianjin, China, in 2000, and M.A.Sc and Ph.D degrees both in Electrical Engineering from Carleton University, Ottawa, ON, Canada, in 2003 and 2008, respectively. She is currently a RF modeling engineer at Freescale Semiconductor.

Her research interests include the development of advanced modeling techniques based on neural networks and space mapping for microwave device modeling and nonlinear circuit behavior modeling, and their applications in computer-aided design for electronic circuits/systems. Her current work involves EM and nonlinear modeling for LDMOS device. She has over 20 technical papers and articles accepted and published in journals, conference proceedings, and international workshop notes.

Qi-Jun Zhang (S'84-M'87-SM'95-F'06) received the B.Eng. degree from the East China Engineering Institute, Nanjing, China, in 1982, and the Ph.D. degree in electrical engineering from McMaster University, Hamilton, ON, Canada, in 1987.

From 1982 to 1983, he was with the System Engineering Institute, Tianjin University, Tianjin, China. From 1988 to 1990, he was with Optimization Systems Associates (OSA) Inc., Dundas, ON, Canada, where he developed advanced microwave optimization software. In 1990, he joined the Department of Electronics, Carleton University, Ottawa, ON, Canada, where he is currently a Professor. He has authored over 200 publications. He authored Neural Networks for RF and Microwave Design (Artech House, 2000) and coedited Modeling and Simulation of High-Speed VLSI Interconnects (Kluwer, 1994). He contributed to Encyclopedia of RF and Microwave Engineering, (Wiley, 2005), Fundamentals of Nonlinear Behavioral Modeling for RF and Microwave Design (Artech House, 2005), and Analog Methods for Computer-Aided Analysis and Diagnosis (Marcel Dekker, 1988). He was a Guest Co-Editor for the "Special Issue on High-Speed VLSI Interconnects" of the International Journal of Analog Integrated Circuits and Signal Processing (Kluwer, 1994) and was a two-time Guest Editor for the "Special Issue on Applications of ANN to RF and Microwave Design" of the International Journal of RF and Microwave CAE (Wiley, 1999, 2002). He is an Editorial Board member of the International Journal of RF and Microwave Computer-Aided Engineering and the International Journal of Numerical Modeling. His research interests are neural network and optimization methods for high-speed/high-frequency circuit design.

Dr. Zhang is a member on the Editorial Board of the IEEE TRANSACTIONS ON MICROWAVE THEORY AND TECHNIQUES. He is a member of the Technical Committee on Computer-Aided Design (MTT-1) of the IEEE Microwave Theory and Techniques Society (IEEE MTT-S).

Tom Dhaene (M'94-SM'05) was born in Deinze, Belgium, on June 25, 1966. He received the Ph.D. degree in electrotechnical engineering from the University of Ghent, Ghent, Belgium, in 1993. From 1989 to 1993, he was a Research Assistant with the Department of Information Technology, University of Ghent, where his research focused on different aspects of full-wave electromagnetic (EM) circuit modeling, transient simulation, and time-domain characterization of high-frequency and high-speed interconnections. In 1993, he joined the EDA company Alphabit (now part of Agilent Technologies). He was one of the key developers of the planar EM simulator ADS Momentum, and he is the principal developer of the multivariate EM-based adaptive metamodeling tool ADS Model Composer. He was a Professor with the Computer Modeling and Simulation (COMS) Group, Department of Mathematics and Computer Science, University of Antwerp, Antwerp, Belgium. He is currently a Full Professor with the Department of Information Technology, Ghent University, Ghent, Belgium. He has authored or coauthored over 100 peer-reviewed papers and abstracts in international conference proceedings, journals, and books.