A Study on Global and Local Optimization Techniques for TCAD Analysis Tasks

Thomas Binder, Clemens Heitzinger, and Siegfried Selberherr, *Fellow, IEEE*

***Abstract*—We evaluate optimization techniques to reduce the necessary user interaction for inverse modeling applications as they are used in the technology computer-aided design field. Four optimization strategies are compared. Two well-known global optimization methods, simulated annealing and genetic optimization, a local gradient-based optimization strategy, and a combination of a local and a global method. We rate the applicability of each method in terms of the minimal achievable target value for a given number of simulation runs and in terms of the fastest convergence. A brief overview over the three used optimization algorithms is given. The optimization framework that is used to distribute the workload over a cluster of workstations is described. The actual comparison is achieved by means of an inverse modeling application that is performed for various settings of the optimization algorithms. All presented optimization algorithms are capable of evaluating several targets in parallel. The best optimization strategy that is found is used in the calibration of a model for silicon self-interstitial cluster formation and dissolution.**

***Index Terms*—Inverse modeling, microelectronics, optimization techniques, semiconductors, simulation.**

I. Introduction

The calibration of technology computer-aided design (TCAD) simulators is a tedious task that requires not only a lot of CPU power, but also a certain amount of user interaction. In this paper, an approach to minimize user interaction for inverse modeling tasks is presented. We use a *framework* to utilize a cluster of workstations. Existing solutions are usually designed with gradient-based optimization techniques to find a suitable optimum. This requires user interaction whenever a local optimum is encountered. In this paper, we compare a local, two global, and a combined local-global optimization technique for the purpose of inverse modeling. Two examples were conducted. In the first step, a benchmarking example, the inverse modeling of a long channel NMOS device, is used to compare the different optimization techniques. In a second experiment, the best strategy found in the benchmark is used to calibrate a diffusion model of a commercial TCAD simulator.

A. Optimizers

In the benchmarking application, four different strategies are rated. These are a local, gradient-based optimization technique, a genetic-optimization technique, optimization by simulated annealing, and a combination of simulated annealing with the local optimization strategy. In the following, a brief overview over the used algorithms is presented.

1) Local Optimizer: Gradient-based optimization strategies iteratively search for a minimum of a D dimensional target function $f(\vec{x})$. The target function is thereby approximated by a terminated Taylor series expansion around $\vec{x_0}$

$$f(\vec{x}_0+\vec{x}) \approx f(\vec{x}_0) + \left(\nabla f(\vec{x}_0)\right)^{\mathcal{T}}\vec{x} + \frac{1}{2}\vec{x}^{\mathcal{T}}\nabla^2 f(\vec{x}_0)\vec{x}.$$

Levenberg-Marquardt Optimizer: The Levenberg-Marquardt [1] optimizer (LMMIN) uses a so-called trust-region method, where the search direction $\vec{p}$ is a combination of steepest descent and the Newton direction. The search direction is defined by

$$\vec{p}(\boldsymbol{J}(\vec{x})^{\mathcal{T}}\boldsymbol{J}(\vec{x})+\lambda\boldsymbol{I}) = -\boldsymbol{J}(\vec{x})^{\mathcal{T}}\vec{f}(\vec{x})$$

where $\boldsymbol{I}$ denotes the unity matrix, $\lambda \geq 0$ the *Marquardt* parameter, and $\boldsymbol{J}(\vec{x})$ the Jacobian matrix

$$\boldsymbol{J}(\vec{x}) = \begin{pmatrix} \frac{\partial f_1(\vec{x})}{\partial x_1} & \cdots & \frac{\partial f_1(\vec{x})}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial f_m(\vec{x})}{\partial x_1} & \cdots & \frac{\partial f_m(\vec{x})}{\partial x_n} \end{pmatrix}.$$

The value of λ is adjusted based on the last evaluation. For a value of $\lambda = 0$, the direction $\vec{p}$ results in the Newton direction, whereas for $\lambda = \infty$ the direction $\vec{p}$ is parallel to the one of the steepest descent method.

The implementation used in the presented examples is based on the MINPACK project [2], [3].

2) Genetic Optimization: Genetic algorithms (GA) were first introduced by Holland [4]. GAs are so-called population-based search strategies. They maintain a set of points, so-called genomes, in a function space and try to make use of analogies to biological evolution by performing mutation and crossover operations between the individuals of a population. Starting with an initial population, the algorithm evolves by iteratively creating a new generation of individuals based on an already existing one. New individuals are introduced by a so-called crossover operation, where at least two individuals of a generation are chosen as parents.

The genetic optimizer used in the presented example was implemented with the C++ library GALIB [5].

3) Optimization by Simulated Annealing: Simulated annealing is a stochastic computational method for finding global extrema to large optimization problems. It was first proposed as an optimization technique by Kirkpatrick in 1983 [6] and Ĉerny in 1984 [7]. The optimization problem can be formulated as a pair of (C, f), where C describes a discrete set

Manuscript received April 6, 2003; revised August 29, 2003. This paper was recommended by Associate Editor S. Saxena.

The authors are with the Institute for Microelectronics, Technical University of Vienna, 1040 Vienna, Austria (e-mail: binder@iue.tuwien.ac.at).

Digital Object Identifier 10.1109/TCAD.2004.828130

of configurations (i.e., parameter values) and f is the objective function that is to be optimized. The problem is then to find a vector $\{C_{\text{opt}}\}$ such that $f(\{C_{\text{opt}}\})$ is optimal.

The optimization algorithm is based on a physical annealing analogy. Physical annealing is a process in which a solid is first heated until all particles are randomly arranged in a liquid state, followed by a slow cooling process. At each (cooling) temperature enough time is spent for the solid to reach thermal equilibrium, where energy levels follow a Boltzmann distribution. As temperature decreases the probability tends to concentrate on low energy states. Care must be taken to reach thermal equilibrium prior to decreasing the temperature. At thermal equilibrium, the probability that a system is in a macroscopic configuration i with energy E_i is given by the Boltzmann distribution.

The behavior of a system of particles can be simulated using a stochastic relaxation technique developed by Metropolis *et al.* [8]. Starting at time t and configuration q, a candidate configuration r for the time $t+1$ is generated randomly. The new candidate is accepted or rejected based on the difference between the energies associated with states q and r. The condition for r to be accepted is determined by

$$p = \frac{p_r}{p_q} = \exp\left(-\frac{E_r - E_q}{kT}\right) > 1. \tag{1}$$

If $p \leq 1$ then r is accepted with probability p. It was shown that for $t \longrightarrow \infty$, the probability that the system is in configuration s equals p_s [9]. One feature of the Metropolis algorithm is that a transition out of a local minimum is always possible at nonzero temperature. Another evenly interesting property of the algorithm is that it performs a kind of adaptive divide and conquer. Gross features of the system appear at higher temperatures, fine features develop at lower temperatures.

For our applications we used the implementation by Ingber [10].

B. Optimization Framework

To carry out large numbers of independent simulations it is well established to use *frameworks* that distribute the work load over a cluster of workstations. In order to perform our experiments, we used the simulation environment SIESTA [11], [12] which features a sophisticated job farming facility. To choose among all available hosts the so-called guess load is used

$$l_{\text{guess}} = l_{\text{base}} + l_{\text{sys}} + \underbrace{\sum_{i=1}^{N} e^{-(t-t_i^{\text{start}})/\tau_{\text{start}}}}_{\text{started jobs}} - \\ - \underbrace{\sum_{i=1}^{M}\left(\left(1 - e^{-(t_i^{\text{start}}-t_i^{\text{stop}})/\tau_{\text{stop}}}\right)\cdot e^{-(t-t_i^{\text{stop}})/\tau_{\text{stop}}}\right)}_{\text{stopped jobs}}. \tag{2}$$

This is an estimation of the actual load which is computed by superimposing the system load (l_{sys}), the number of jobs already running (N), and the number of recently finished jobs (M). The base load (l_{base}) is used to reserve computing resources for jobs that are running outside SIESTA. The time constants τ_{start} and τ_{stop} are necessary to account for the actual time that elapses between the start of a job or its termination respectively and the reflection in the system load. These parameters are architecture dependent and must be determined experimentally. An empirically determined value for a LINUX system is 5 s. Based on the guess load, the rank R_i of a host is computed

$$R_i = \frac{\max\left[\frac{l_{\text{guess}}+1}{n_i^{\text{cpu}}}, 1\right]}{s_i}$$

where n_i^{cpu} is the number of CPUs, and s_i is the relative speed of a host. The rank is finally used to select a host.

All optimization algorithms are integrated into SIESTA by means of a protocol. SIESTA starts an optimizer as a separate process and communicates with this process via standard input and standard output. This makes it possible to switch easily between the different optimizers.

II. BENCHMARKING OF OPTIMIZATION ALGORITHMS

To compare the different optimization strategies, a reasonably simple example was chosen in order to keep the total CPU time low. In this benchmark, we used a simple schematic NMOS transistor device, where the doping profiles are assumed to follow Pearson Type IV and Gaussian type distributions. No process simulation was carried out to create the device, instead a small program that uses as input the parameters for the distribution functions was used to create the device for the device simulator MINIMOS-NT [13], [14]. The resulting $I_D V_D$ and $I_D V_G$ curves were compared against measurements, the deviation was used as optimization target.

All optimizers need a target function to rate the error of computed and measured data points. The target function used in the *framework* is

$$t_{\text{opt}} = \sqrt{\frac{1}{\sum\limits_{j=1}^{n} N_j} \cdot \sum_{j=1}^{n}\sum_{i=1}^{N_J} S(\text{pm}_{ji}, \text{pc}_{ji})^2} \tag{3}$$

where pc_j and pm_j are points of computed and measured curves respectively, N_j is the dimension of a curve, n is the total number of curves, and S is a scaling function used to keep the target value within certain limits.

The benchmark was performed by extracting the profile of the NMOS device with all different optimization algorithms. Fig. 1 depicts a schematic block diagram of the *inverse modeling* task. All optimizations were compared after a total of 700 evaluations, where the average simulation time[1] was $\approx$ 1 min.

A. Device Model

The doping profile was extracted from an artificial NMOS device with a channel length of 9 μm. This particular device was selected in order not to have to account for parasitic effects in the device simulation. This is justified since the device is only used as a benchmark. Fig. 2 shows the two-dimensional

[1]The simulation time in this example depended on the input device. For parameter sets close to the found optimum the simulation time was below 1 min, whereas for "bad"parameter sets the simulation times increased. In some cases the device simulator did not converge and had to be killed after a timeout period of 5 min.

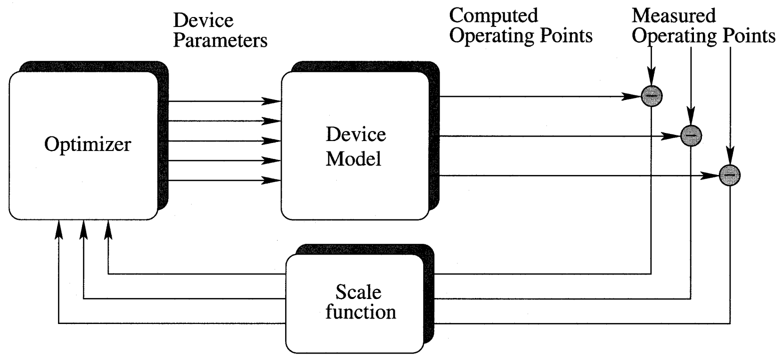

Fig. 1. *Inverse modeling* of doping profiles.

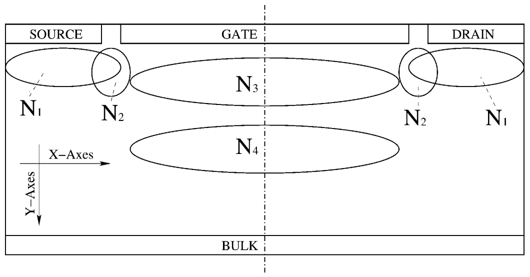

Fig. 2. 2-D device model with analytical doping peaks.

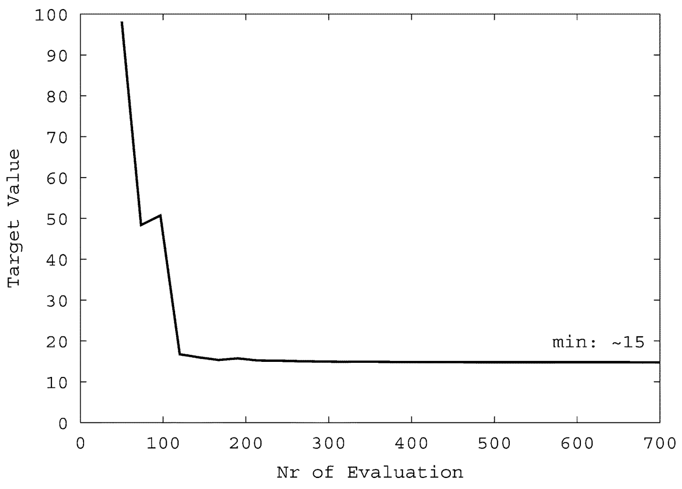

Fig. 3. Progress of the gradient-based optimizer.

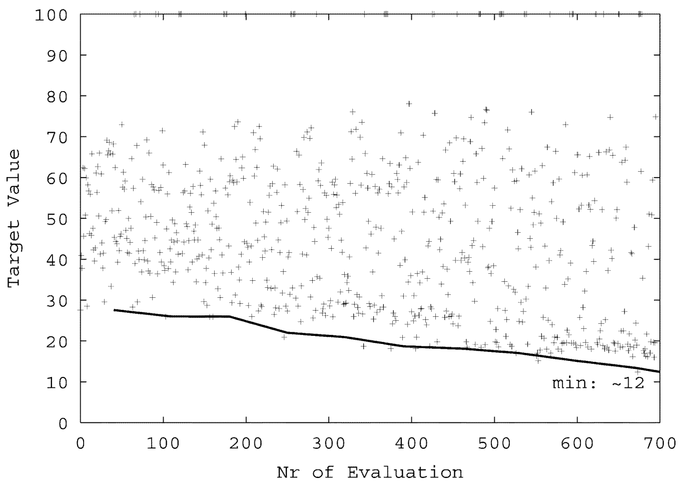

Fig. 4. Evolution of the genetic optimizer for $P_{\text{cross}} = 0.9, P_{\text{mut}} = 0.2$, and two-point crossover.

(2-D) model of the device under consideration. The device is symmetric along the dash-dotted line. The elliptically shaped regions denote the analytical dopant concentrations. The dopant profiles are approximated by Pearson Type IV (peaks N_1–N_3), and Gaussian (peak N_4) functions as described in [15]. The Gaussian function is qualified by the parameters projected range R_p, standard deviation σ, and the peak value N. Pearson Type IV functions additionally define skewness (γ) and kurtosis (β) and σ in y and x direction. For the source/drain doping peaks (N_1 and N_2) an extra parameter to define the x coordinate of the peak is defined. This results in seven parameters for the source/drain peaks, five parameters for N_3 and three parameters for N_4. Other parameters are the oxide thickness t_{ox} the delta in the gate length Δ_{lg} and the donor and acceptor background dopings. A total of 26 free parameters was optimized. All used optimizers were capable of approximating the given dopant profile within few percent of deviation.

B. Local Optimizer

The performance of gradient-based methods strongly depends on the initial values supplied. Several optimization runs with different *initial guesses* might be necessary if no *a priori* knowledge (e.g., the result of a process simulation) about the dopant concentration profile is available. Fig. 3 shows the evolution of the target values for a certain *initial guess*. In this example, the optimizer was stopped at a local minimum.

C. Genetic Algorithms

The genetic optimizer delivered the best results with the STEADY-STATE algorithm, the ROULETTE-WHEEL selection scheme, and the one- and two-point crossover methods, respectively. We used a replacement percentage of $P_{\text{replace}} = 70.0\%$ and a population size of 40. Since GALIB does not directly support parallel target evaluation our optimizer takes care of evaluating several jobs in parallel. Several experiments with different crossover and mutation probabilities were carried out. Fig. 4 depicts the best settings for crossover and mutation that were found. Note that the best individual within a population sometimes occurs at a lower evaluation number thus appearing below the solid line.

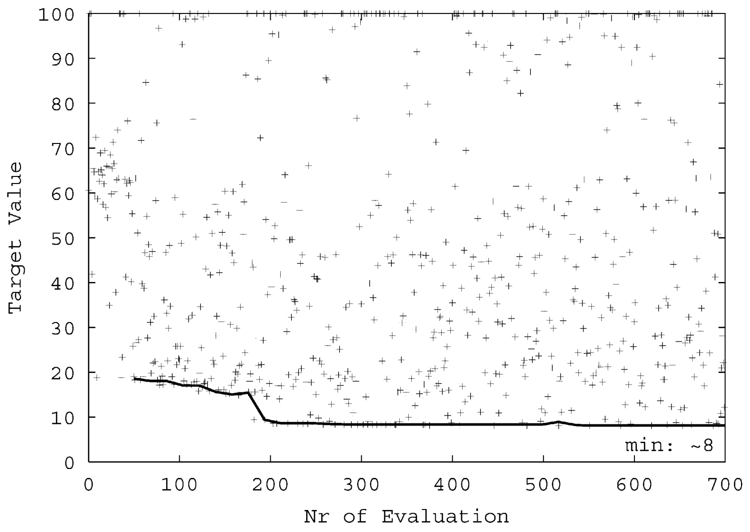

Fig. 5. Evolution of the *simulated annealing* optimizer. The best target value of ≈ 8 was already reached after ≈ 200 evaluations.

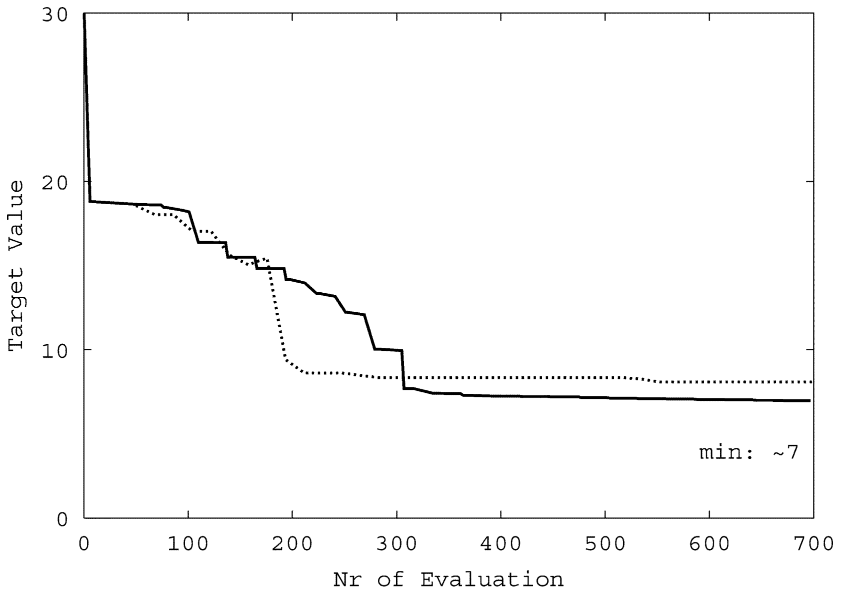

Fig. 6. Combination of simulated annealing and the local optimizer. The first ≈ 70 evaluations where performed by simulated annealing, the rest was evaluated with the local optimizer. The best target value from the standalone simulated annealing run (dotted line) was reached after ≈ 300 evaluations.

D. Simulated Annealing

Fig. 5 shows the progress of the simulated annealing algorithm. Standard parameter settings were used. Compared to the genetic algorithm this optimizer reaches the same target value within approximately one third of evaluations. The original software was extended to support the parallel evaluation of simulation jobs. The number of parallel running jobs is varied between configurable bounds. As long as no improvement of the target value is found the number of parallel jobs is increased up to a configurable maximum, otherwise this number is decreased down to a configurable minimum. The minimum and maximum values are configured to best utilize the number of available CPUs.

E. Combined Optimization Algorithm

The promising results obtained in the comparison of the global optimization techniques seem to justify a combination of simulated annealing and the local optimizer. It was tried to: 1) achieve the best target value obtained from simulated annealing (≈ 8) with the least possible number of evaluations and 2) to achieve a better target value with the given number of fixed evaluations.

Fig. 6 depicts a run where the local optimizer was started after ≈ 70 evaluations of simulated annealing, and performed the rest of the whole optimization. As *initial guess* for the local optimization run the best target of the first evaluations that were performed so far was used. For reference, the evolution of the standalone simulated annealing run is plotted as a dash-dotted line (which corresponds to the solid line of Fig. 5).

Fig. 7 depicts a strategy that reaches the best target of the standalone simulated annealing run even faster. In this experiment, the global and local algorithms are run alternately with the global optimizer starting. The best target of one optimizer is thereby taken as *initial guess* for the other optimizer respectively. In the very first simulated annealing run 100 evaluations

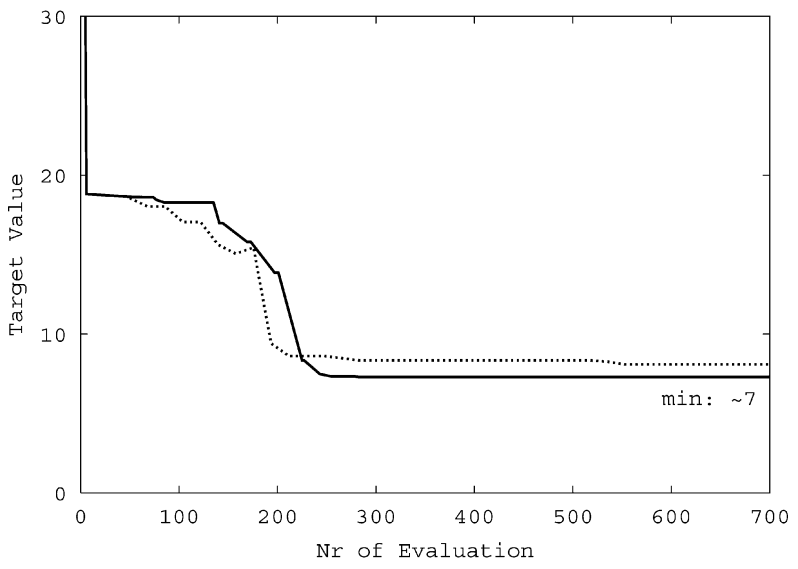

Fig. 7. Better combination of simulated and the local algorithm. The optimizers are started alternately. The simulated annealing algorithm performed the first ≈ 100 initial evaluations, then a local optimization was started after each 30 simulated annealing evaluations. The local optimization was terminated as soon as the target was improved. With this strategy, the target from the standalone simulated annealing run was already reached after ≈ 230 evaluations.

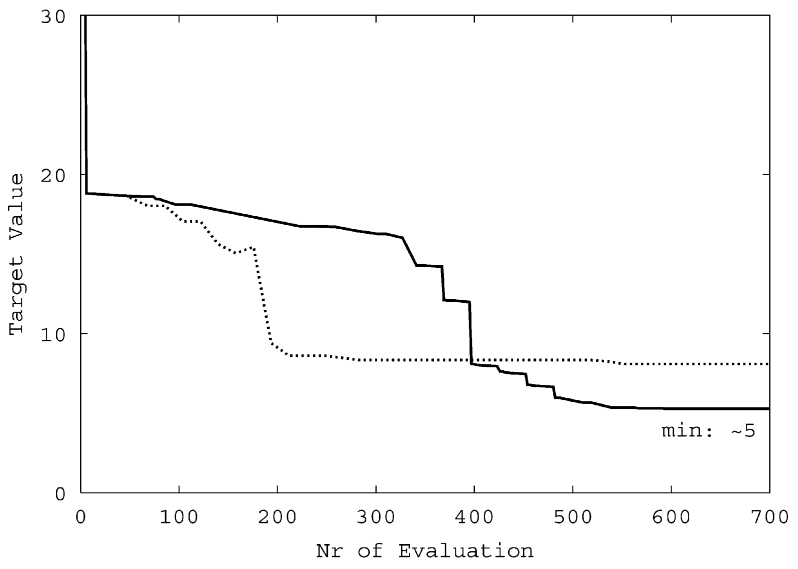

Fig. 8. This combination strategy of simulated annealing and the local optimizer results in the best target value for the given maximum number of evaluations. The first 70 initial evaluations were performed with simulated annealing, then a local optimization run was invoked after every 40 evaluations of simulated annealing.

were performed then the local optimizer was started. As soon as an improvement in the target value was detected, the local optimization was terminated and the global was continued with its best state updated accordingly. After 30 evaluations the local optimizer was invoked again.

The strategy depicted in Fig. 8 results in the best target value found in all experiments carried out with the combined optimization technique. Here, 70 initial simulated annealing evaluations were performed and 40 evaluations were performed between two runs of the local optimizer. This strategy also reached the best target value of ≈ 3.8 among all experiments. This target value was already reached after ≈ 800 evaluations and was not improved further.

The combination of global and local optimization techniques results in a very robust optimization algorithm. The fact of longer optimization times for global optimization techniques is at least partly compensated for by using parallelizable algorithms. Only minimal interaction by the user is necessary during the optimization run. This justifies the usage of this algorithm for the following application.

III. Calibration of a TCAD Simulator

In this section, an industrial application, the calibration of a model for silicon self-interstitial cluster formation and dissolution is given. The formation and dissolution of silicon self-interstitial clusters is attributable to transient enhanced diffusion (TED). TED is the fast redistribution of impurities that takes place in the very first thermal step right after implantation. An accurate simulation of TED plays an important role in the manufacturing process of submicron semiconductor devices [16], [17]. A well-calibrated model is thereby a prerequisite for an accurate simulation of TED.

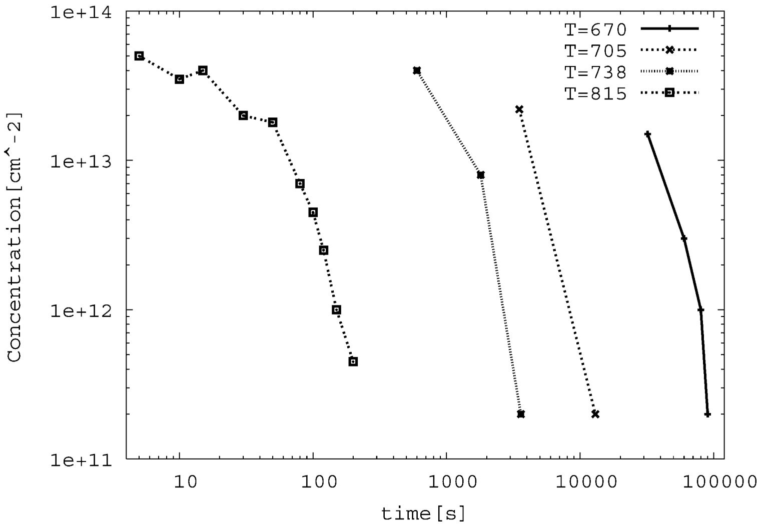

Fig. 9. Silicon self-interstitial density (stored in {113} or {311} defects) in cm^{-2} as a function of time for different annealing temperatures. The implants were performed with a dose of $5 \cdot 10^{13}$ cm^{-2} and at an energy of 40 *keV*.

In [18], the source for the silicon self-interstitials was identified to be {113} defects which are rod like clusters of interstitials. Counting the number of interstitials is a complicated task. Transmission electron microscopes are used to measure the number of interstitials in each defect [19]. In this application the one moment interstitial clustering model of the TSUPREM-IV [20] simulator is calibrated to a set of measurements published in [19]. The TSUPREM-IV process simulator is part of the Synopsys TCAD suite, originally offered by a company named TMA. The products of TMA, however, were first incorporated by Avant!, and later by Synopsis [21]. Our simulation example is comprised of an analytical ion implantation followed by the actual diffusion that is to be calibrated. No complex process simulations were performed. The CPU time for a single simulation run was $\approx 2 - 3$ min.

The measurements were carried out at temperatures 670, 705, 738, and 815°C. Fig. 9 depicts the measured curves at the different temperatures.

A. Model for Cluster Formation and Dissolution

In [16], the equation describing interstitial cluster kinetics is given as

$$\frac{\partial C}{\partial t} = 4\pi \alpha a D_I I C - \frac{C D_I}{a^2} e^{-E_b/kT} \tag{4}$$

where a denotes the interatomic spacing, α the capture radius expressed in units of the interatomic spacing, $D_I = D_0 e^{-E_m/kT}$ the interstitial diffusivity, $I(t,x)$ the concentration of unclustered interstitials, $C(t,x)$ the concentration of clustered interstitials, t the time, T the annealing temperature in Kelvin, and k the Boltzmann constant. TSUPREM-IV uses a general clustering model that uses many of the models proposed in literature as subsets. The main formula for the change of the concentration of clustered interstitials is [20]

$$\frac{\partial C}{\partial t} = K_{fi} \frac{I^{ifi}}{I_*^{isfi}} + K_{fc} \frac{I^{ifc}}{I_*^{isfc}} (C + \alpha I)^{cf} - K_r C^{\mathrm{cr}} \tag{5}$$

TABLE I
PARAMETERS AS THEY WERE USED IN THE CALIBRATION OF THE CLUSTERING MODEL, AND THE FOUND OPTIMAL VALUE

Parameter	TSUPREM-IV Variable	default value	optimum
D_0	D.0	51.0	694.335
$kfi0$	CL.KFI.0	$5.0 \cdot 10^{24}$	$6.3213 \cdot 10^{26}$
$kfiE$	CL.KFI.E	3.774	3.91624
$kfc0$	CL.KFC.0	$4.368 \cdot 10^{19}$	$6.95285 \cdot 10^{19}$
$kfcE$	CL.KFC.E	4.95	4.90781
$kr0$	CL.KR.0	$2.8 \cdot 10^{16}$	$1.42833 \cdot 10^{17}$
krE	CL.KR.E	3.57	3.61939
α	CL.KFCI	1100	1336
cr	CL.CR	1.0	0.982801
cf	CL.CF	1.0	1.02736

TABLE II
PARAMETER VALUES AS USED IN THE CALIBRATION EXAMPLE, THEIR RANGES, AND UNITS

Parameter	minimum value	maximum value	unit
D_0	25.0	1000.0	$cm^2 s^-1$
$kfi0$	$1.0 \cdot 10^{20}$	$1.0 \cdot 10^{28}$	$cm^{-3(1+\mathrm{isfi}-\mathrm{ifi})} s^-1$
$kfiE$	3.4	6.0	eV
$kfc0$	$1.0 \cdot 10^{17}$	$7.0 \cdot 10^{19}$	$cm^{-3(1+\mathrm{isfc}-\mathrm{ifc}-\mathrm{cf})} s^-1$
$kfcE$	4.9	5.2	eV
$kr0$	$1.5 \cdot 10^{16}$	$1.0 \cdot 10^{18}$	$cm^{-3(1-cr)} s^-1$
krE	3.0	3.62	eV
α	0	5000	1
cr	0.98	8.0	1
cf	0.5	1.03	1

where $I_*(t,x)$ denotes an equilibrium concentration of interstitials. All other symbols of (5) are parameters that need to be adjusted. K_{fi}, K_{fc}, and K_r are the reactions constants which have the form

$$K_{fi} = kfi0 \cdot e^{-kfiE/kT} \tag{6}$$

$$K_{fc} = kfc0 \cdot e^{-kfcE/kT} \tag{7}$$

$$K_r = kr0 \cdot e^{-krE/kT} \tag{8}$$

with coefficients $kfi0$, $kfc0$, and $kr0 > 0$. Since the coefficients are all positive, the first two terms of (5) describe

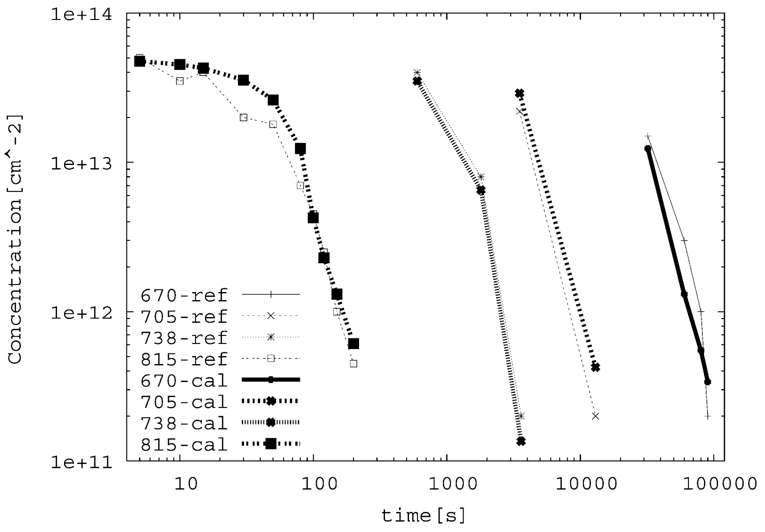

Fig. 10. Deviation of measured and computed cluster concentration (with the calibrated model) in cm^{-2}. The reference concentrations are drawn in thin lines.

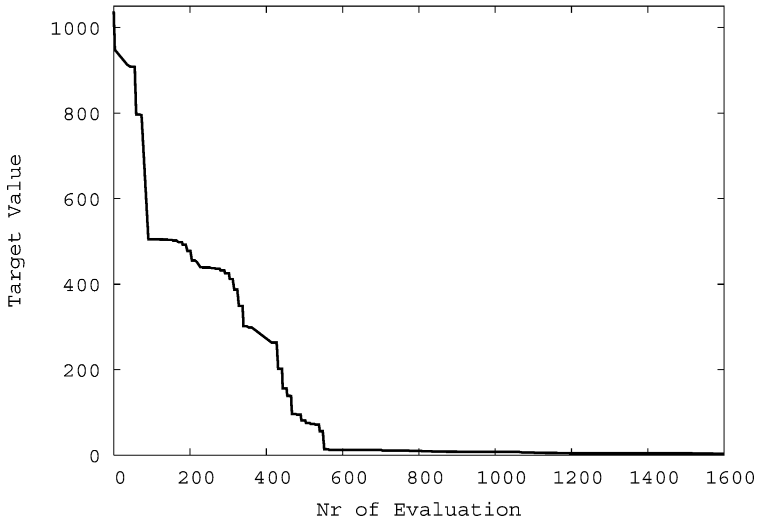

Fig. 11. Progress of combined optimizer. The plot shows the first 1600 evaluations. The optimizer started at a target value of ≈ 1050.

the formation of clusters, whereas the last term describes its dissolution. The term $K_{fi}(I^{ifi}/I_*^{isfi})$ describes the joining of two clusters[2]. Therefore, the parameter values ifi and $isfi$ are expected to be 2. The second term, $K_{fc}I^{ifc}/I_*^{isfc}\,(C+\alpha I)^{cf}$, describes the case where an unclustered interstitial joins a cluster. A value of 1 for ifc and $isfc$ is assumed.

B. Calibration Task

The calibration of the TSUPREM-IV parameters as described above was performed with the combination of the simulated annealing and the gradient based optimization algorithm. Note that two measured points were ignored in this experiment since their given concentration is above the implanted dose. The TSUPREM-IV manual [20] suggests a value of 1.0 for the parameter cr, however, to further confirm the chosen optimization method this parameter was also optimized.

Tables I and II depict all parameters, their ranges and units, and the found optimum. All shown parameters were optimized in the calibration run.

[2]The ratio I/I_* is called interstitial supersaturation.

To account for the range of ≈ 3 decades of the dopant concentration and under the assumption that all computed concentrations assume positive values, the following scale function $S(p)$ was used to compute the deviation of a computed from a measured point:

$$S(p_m,p_c)=100\cdot\left(10^{|(\log_{10}p_m-\log_{10}p_c)/\max[\log_{10}p_m,\log_{10}p_c]|}-1\right) \tag{9}$$

where p_m denotes a measured concentration and p_c denotes a computed one (delivered by TSUPREM-IV). The error vector is computed according to (3). The obtained (optimized) parameter values are shown in Table I. Fig. 10 depicts the resulting deviation of computed from measured concentrations for the model parameters given in column optimum of Table I. Figs. 11 and 12 depict the progress of the combined (simulated annealing and local) optimizer. The optimization started with a target value of $t_{\text{opt}} = 1037$. The best target of 3.44 was reached after $\approx$ 1600 evaluations. The optimization was stopped after a total of 3300 evaluations. No improvement in the target value was found for evaluation numbers 1600 – 3300.

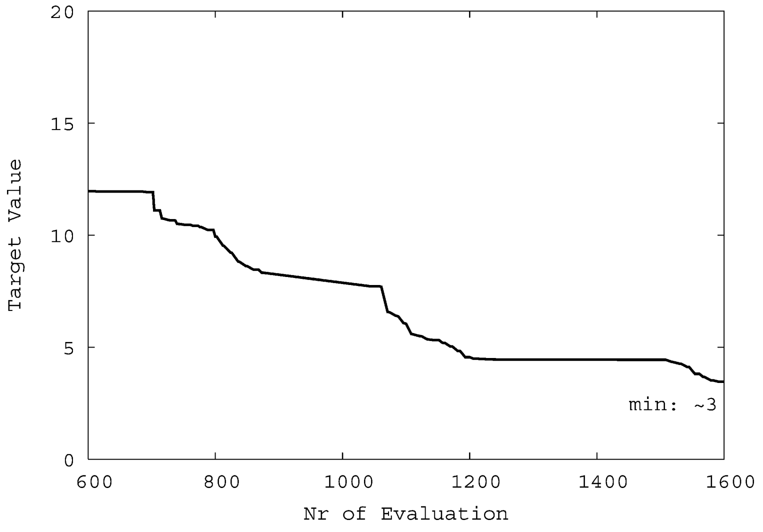

Fig. 12. This plot depicts a detailed view of Fig. 11. Evaluation numbers 600 – 1600 are shown. The final optimum of 3.44 was reached after 1600 evaluation. No further improvement was found within another 1700 evaluations.

The found optimum target value of ≈ 3 fully supports the used optimization strategy. The value of 1.0 found for cr agrees very well with the value suggested in the TSUPREM-IV manual. The values found for cr, $kfc0$, $kfcE$, and krE are very close to their selected bounds. However, the bounds were chosen deliberately by taking into account the following facts: According to the TSUPREM-IV manual, a too small value for cr (which describes the dissolution of clusters) might result in numerical instabilities of the simulation. Parameters $kfcE$, and krE denote energies, they exponentially impact the reaction. Too high values for those parameters would result in a nonphysically fast diffusion. Finally, the bounds for $kfc0$ were chosen in conjunction with $kfcE$ to avoid simulation instabilities.

It is worth mentioning that the usage of a cluster of workstations (as opposed to using a single machine) drastically reduces the wall clock time that is spent in the optimization task. Both, gradient and global optimizers fully utilize the cluster. The total time reduction can be estimated to be directly related to the sum of the used CPU speeds (in MHz). The communication overhead is typically very low since a target evaluation (simulation run) usually takes several minutes to compute.

IV. Conclusion and Outlook

We presented four different optimization strategies suitable for *inverse modeling* tasks in the TCAD field. These are a local, two global, and a combination of a local and a global optimization technique. The strategies were compared by means of an *inverse modeling* task where the doping profile of an artificial NMOS transistor was identified. The optimization *framework* that is used to distribute the workload over a cluster of workstations was presented. An industrial application, the calibration of a model for self-interstitial cluster formation and dissolution was given.

The local gradient based method is the fastest if the *initial guess* is chosen appropriately but stops in a local minimum or even fails to converge otherwise. In this case, the whole optimization must be restarted with a different *initial guess*. On the other hand, the presented global optimization techniques demonstrate very robust optimization strategies. The necessary interaction by a user is reduced to set up the experiment. There is no need to supply an *initial guess*. The tradeoff is, clearly, the extra CPU power that is consumed by the global algorithms. This, however, can be partly compensated by a better scalability of the optimization problem to a cluster of workstations.

For the case of the genetic optimizer, the crossover and mutation probabilities are very crucial to the performance. It turned out to be rather time consuming to find acceptable values for those parameters. The simulated annealing algorithm on the other hand did not ask for any parameter tuning at all. Standard settings for all parameters were used. Among the two global optimization strategies this optimizer clearly demonstrates the better performance.

The combination of simulated annealing with the local algorithm gave the best performance for the benchmarking example. It is both faster and delivers a better optimum then any other optimization strategy we tried. It effectively combines the good convergence of local optimization techniques with the robustness of the global technique. This is in good agreement with the work of Desal *et al.* [22]. The combined optimizer was used to calibrate a diffusion model of the commercial process simulator TSUPREM-IV . User interaction is reduced to setting up the experiment.

An optimization of the combined optimization strategy that tries to minimize the overall number of simulation runs by properly choosing the number of simulations that are run in global and local mode respectively is not easy. One could try to start one optimization on top of another, however, this would increase the total evaluations exponentially. Therefore, this approach does not seem feasible. Instead, only a few manually chosen combinations were tried.

If a further reduction of necessary user interaction is desired, the *framework* must be user friendly in setting up optimization experiments. Our current implementation uses a LISP like syntax to define the models and their interaction. This is a powerful means to describe almost any possible optimization problem at hand, but it is still not very well accepted by the user. Additionally, it has proven too complicated for being used on a regular basis. As a solution, one could think of a tool that generates model descriptions suitable for SIESTA based on a choice of selectable models. In a library, the tool can store several frequently used models (e.g., calibration task with TSUPREM-IV, calibration with MINIMOS-NT, etc.). This would greatly reduce the time spent in setting up at least the supported experiments by still keeping the flexibility of the description language.

Acknowledgment

The authors would like to thank the Christian Doppler Forschungsgesellschaft and the Sony Corporation, Hon-Atsugi, Japan, for their support.

References

[1] D. W. Marquardt, "An algorithm for the estimation of nonlinear parameters," *Soc. Ind. Appl. Math. J.*, vol. 11, pp. 431–441, 1963.

[2] J. J. Moré, D. C. Sorensen, K. E. Hillstrom, and B. S. Garbow, *The MINPACK Project, Sources and Development of Mathematical Software*. Englewood Clifs, NJ: Prentice-Hall, 1984.

[3] J. J. Moré, B. S. Garbow, and K. E. Hillstrom, "Users guide for MINPACK-1," Argonne National Laboratory, Argonne, IL, Tech. Rep. ANL-80-74, 1980.

[4] J. Holland, *Adaption in Natural and Artificial Systems*. Ann Arbor, MI: Univ. of Michigan Press, 1975.

[5] M. Wall. (2000) GAlib a C++ Library of Genetic Algorithm Components. Mass. Inst. of Technol., Cambridge. [Online]. Available: http://lancet.mit.edu/ga

[6] S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[7] V. Ĉerny, "Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm," *J. Opt. Theory Appl.*, vol. 45, pp. 41–45, 1985.

[8] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, and A. H. Teller, "Equation of state calculations by fast computing machines," *J. Chem. Phys.*, vol. 21, no. 6, pp. 1087–1092, 1953.

[9] S. Geman and D. Geman, "Stochastic relaxation, Gibbs distribution and the Bayesian restoration in images ," *IEEE Trans. Patt. Anal. Mac. Int.*, vol. 6, pp. 721–741, Nov. 1984.

[10] L. Ingber, "Very fast simulated re-annealing," *Math. Comput. Model.*, vol. 12, pp. 967–973, 1989.

[11] C. Heitzinger and S. Selberherr, "An extensible TCAD optimization framework combining gradient based and genetic optimizers," presented at the Proc. Int. Symp. Microelectron. Assembly, Singapore, Nov. 2000.

[12] R. Strasser, R. Plasun, and S. Selberherr, "Practical inverse modeling with SIESTA," in *Proc. Simul. Semiconduct. Process. Devices*, Kyoto, Japan, Sept. 1999, pp. 91–94.

[13] T. Grasser, V. Palankovski, G. Schrom, and S. Selberherr, "Hydrodynamic mixed-mode simulation," in *Proc. Simul. Semiconduct. Process. Devices*, Leuven, Belgium, Sept. 1998, pp. 247–250.

[14] T. Binder, K. Dragosits, T. Grasser, R. Klima, M. Knaipp, H. Kosina, R. Mlekus, V. Palankovski, M. Rottinger, G. Schrom, S. Selberherr, and M. Stockinger. (1998) MINIMOS-NT User's Guide. Institut für Mikroelektronik. [Online] Available: http://www.iue.tuwien.ac.at/doku/minimos_docu/mmntdocu.html.

[15] S. Selberherr, *Analysis and Simulation of Semiconductor Devices*. New York: Springer-Verlag, 1984.

[16] C. S. Rafferty, G. H. Gilmer, J. Jaraiz, D. Eaglesham, and H.-J. Gossmann, "Simulation of cluster evaporation and transient enhanced diffusion in silicon," *Appl. Phys. Lett.*, vol. 68, no. 17, pp. 2395–2397, Apr. 1996.

[17] M. Uematsu, "Simulation of clustering and transient enhanced diffusion of boron in silicon," *J. Appl. Phys.*, vol. 84, no. 9, pp. 4781–4787, 1998.

[18] D. J. Eaglesham, P. A. Stolk, H. J. Gossmann, and J. M. Poate, "Implantation and transient B diffusion in Si: The source of interstitials," *Appl. Phys. Lett.*, vol. 65, no. 18, pp. 2305–2307, 1994.

[19] J. M. Poate, D. J. Eaglesham, G. H. Gilmer, J.-J. Gossmann, M. Jaraiz, C. S. Rafferty, and P. A. Stolk, "Ion implantation and transient enhanced diffusion," in *Proc. Int. Electron Devices Meeting*, Washington, DC, 1995, pp. 77–80.

[20] *TSUPREM-4, Two-Dimensional Process Simulation Program, Version 6.5 User's Manual*, SYNOPSYS, Inc., 2003.

[21] SYNOPSYS, Inc.. [Online]. Available: http://www.synopsys.com/products/avmrg/product_flow/technology_cad.html.

[22] R. Desai and R. Patil, "SALO: Combining simulated annealing and local optimization for efficient global optimization," in *Proc. 9th Florida AI Res. Symp.*, June 1996, pp. 233–237.

Thomas Binder was born in Bad Ischl, Austria, in 1969. He received the Diplomingenieur and Doctoral degrees in electrical engineering and computer science at the Technical University of Vienna, Vienna, Austria, in 1996 and 2002.

During his studies he was working on several software projects, mainly in the computer-aided design, geodesy, and security fields. In March 1997, he joined the Institute for Microelectronics, Vienna, Austria, where he is currently working on his doctoral degree. In autumn 1998, he held a visiting research position at Sony, Atsugi, Japan. His scientific interests include data modeling, algorithms, software engineering, and semiconductor technology, in general.

Clemens Heitzinger was born in Linz, Austria, in 1974. He received the Diplom-Ingenieur degree (honors) in technical mathematics and the doctoral degree in technical sciences (honors) from the Technical University of Vienna, Vienna, Austria, in 1999 and 2002, respectively.

From March to May 2001, he was a Visiting Researcher at the Sony Technology Center, Hon-Atsugi, Tokyo, Japan. His scientific interests include process simulation for semiconductor devices and applied mathematics for simulation in microelectronics.

Siegfried Selberherr (M'79–SM'84–F'93) was born in Klosterneuburg, Austria, in 1955. He received the Dipl.-Ing. degree in electrical engineering and the doctoral degree in technical sciences from the Technical University of Vienna, Vienna, Austria, in 1978 and 1981, respectively.

He has held the *venia docendi* on computer-aided design since 1984. Since 1988, he has been Head of the Institute for Microelectronics and, since 1999, has been Dean of the Faculty for Electrical and Information Technology. His current topics are modeling and simulation of problems for microelectronics engineering.