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

T. Binder, C. Heitzinger, and S. Selberherr

Institute for Microelectronics, TU Wien, Gusshausstr. 27–29
A-1040 Vienna, Austria, Thomas.Binder@iue.tuwien.ac.at

ABSTRACT

We compare the two well-known global optimization methods, simulated annealing and genetic optimization, to a local gradient-based optimization technique. We rate the applicability of each method in terms of the minimal achievable target value for a given number of simulation runs in an inverse modeling application.

The gradient-based optimizer used in the experiment is based on the Levenberg-Marquardt algorithm. The actual implementation (lmmin) was taken from MINPACK [1]. The genetic optimizer (genopt) is based on GALIB [2]. For the simulated annealing [3] optimizer (siman) an implementation by L. Ingber was taken. All optimizers are capable of evaluating several targets in parallel.

Keywords: Optimization Techniques, Inverse Modeling, Simulation, Semiconductors, Microelectronics

1 INTRODUCTION

In our inverse modeling experiment the dopant concentration profile of an NMOS transistor should be identified. We use the deviation of computed $I_D V_D$ and $I_D V_G$ curves from measured ones as a target for optimization. The target function as delivered to the optimizer is determined by $\sqrt{(\vec{x} \cdot \vec{x})/N}$ where $\vec{x}$ is the N-dimensional error vector. The error vector is computed as a modified relative error: $100 \cdot (1 - I_c/I_m)$ for $I_c < I_m$ and $100 \cdot (I_m/I_c - 1)$ otherwise [4], where I_c and I_m denote the computed and measured currents, respectively. The dopant profiles are approximated by Pearson Type IV functions as described in [5]. Fig. 1 shows the two-dimensional model of the device under consideration. The elliptically shaped regions denote the analytical dopant concentrations. Fig. 2 shows a plot of the donor and acceptor concentrations and the geometry of a typical device. A total of 27 free parameters was optimized. In order to utilize a cluster of workstations we used our simulation environment SIESTA [6], [7] to distribute the computational load. For the extraction of the curves the device simulator MINIMOS-NT [8], [9] was used.

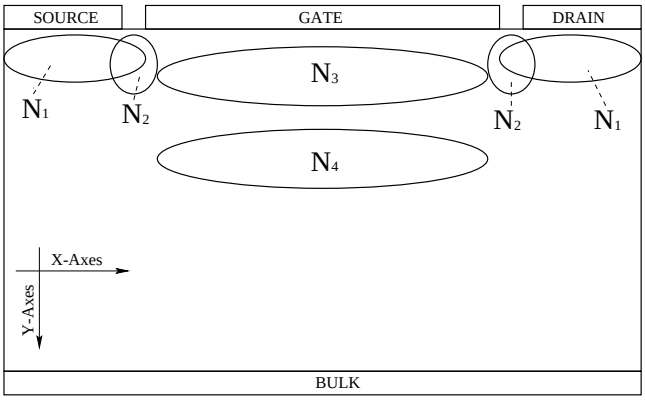

Figure 1: Two-dimensional device model with analytical doping peaks

2 OPTIMIZERS

2.1 Gradient-Based Optimizer

A gradient-based optimizer approximates the target function by a terminated Taylor series expansion:

$$f(\vec{x}_0 + \vec{x}) \approx f(\vec{x}_0) + (\nabla f(\vec{x}_0))^T \vec{x} + \frac{1}{2} \vec{x}^T \nabla^2 f(\vec{x}_0) \vec{x} \quad (1)$$

The actual optimization is performed iteratively. The direction and step width are determined by numerically computing the JACOBIAN and HESSIAN matrices of the target function. Our optimizer uses a finite-difference approximation of the first derivatives thus two evaluations for each parameter are necessary. The second derivatives are computed by using the gradient of the recent and of the last step and the HESSIAN of the last step (Broyden-Fletcher-Goldfarb-Shanno update [10]). The evaluations are independent from each other which means they can be carried out in parallel. The dependence of the number of evaluations on the number of free parameters limits the scalability of the optimizer and thus the utilization of the workstation cluster (for a small number of parameters).

The performance of the 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

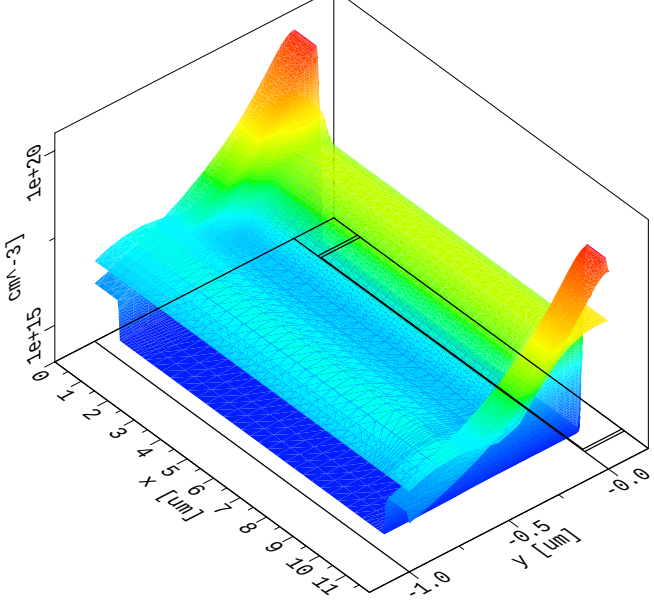

Figure 2: Plot of device with donors and acceptors

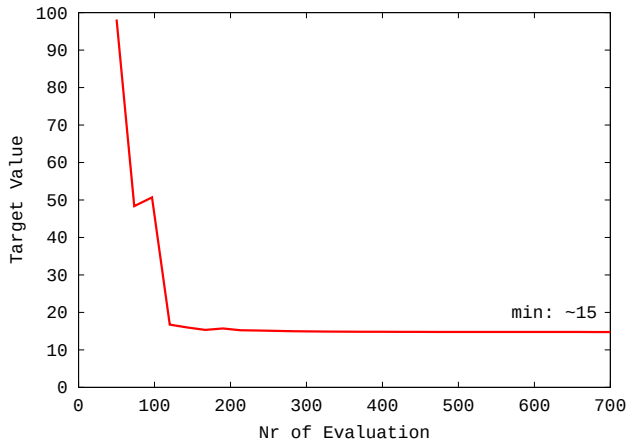

Figure 3: Progress of the gradient-based optimizer

is applied. 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. Care must be taken to provide physically sound bounds for all parameters to avoid simulation failures.

2.2 Genetic Algorithms

Genetic algorithms go back to [11]. A genetic algorithm (GA) is a so called population based search strategy. GA's maintain a set of points (genomes) in a function space. When the optimizer is started an initial population of genomes is chosen. The parameters of the genomes are initialized randomly but within given bounds. The fitness of the individuals in the population is then computed (in our case by means of a device simulation). The simulation result i.e. the target value

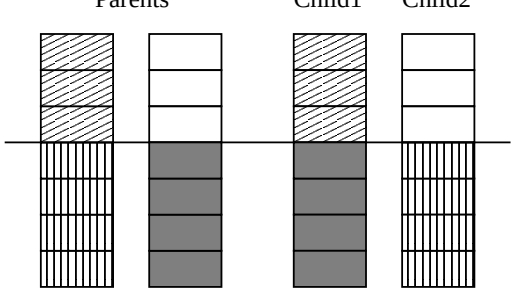

Figure 4: Crossover Operator

is used for selecting individuals for reproduction. The library (GALIB) we used supports four different flavors of genetic algorithms namely SIMPLE (as described in [12]), STEADY-STATE, INCREMENTAL and DEME. They differ in the way individuals are selected for mating, dying and for surviving. In case of the SIMPLE genetic algorithm the whole population is replaced each generation. The STEADY-STATE algorithm replaces only a part of the population. Some of the individuals survive into the next generation. The replacement percentage defines how many individuals are replaced. In the INCREMENTAL algorithm each generation consists of only one or two children. Finally, the DEME algorithm evolves several populations independently each with a STEADY-STATE algorithm. Each generation some individuals are migrated across the populations.

Genetic Reproduction

Reproduction is controlled by mutation and crossover operators. Crossover defines the procedure for generating a child from two parents. The crossover probability (P_{cross}) is used to decide whether the parents or their children are taken over into the next generation. Fig. 4 shows the one-point-crossover method, where a point is chosen randomly to determine which part of the genome to take from mother and father respectively. GALIB supports several crossover methods. For our experiments we used the one-point-crossover and two-point-crossover algorithms. For the optimization task crossover is the attempt to find better individuals by combining the parameters of the best individuals so far.

Mutation introduces new genetic material into a population. Mutation occurs with the probability P_{mut}. One parameter in a genome is replaced by a randomly chosen value (within the allowed range).

Genopt

Our genetic optimizer (genopt) is written using GALIB. For our application we obtained the best results with the STEADY-STATE algorithm. We used a replacement percentage of $P_{replace} = 0.7$ and a population size of 40. Since

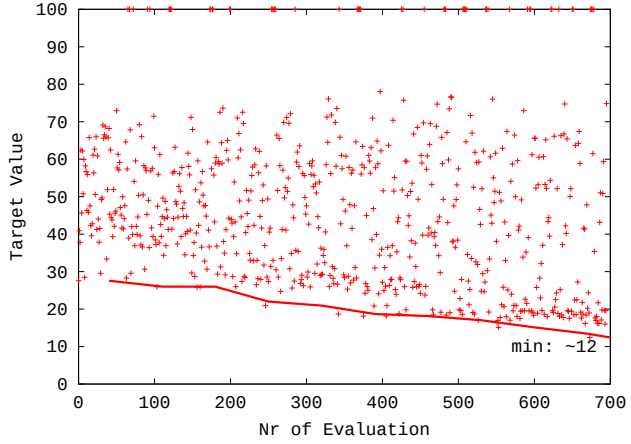

Figure 5: Evolution of the genetic optimizer for $P_{cross} = 0.9$, $P_{mut} = 0.2$ and two-point-crossover

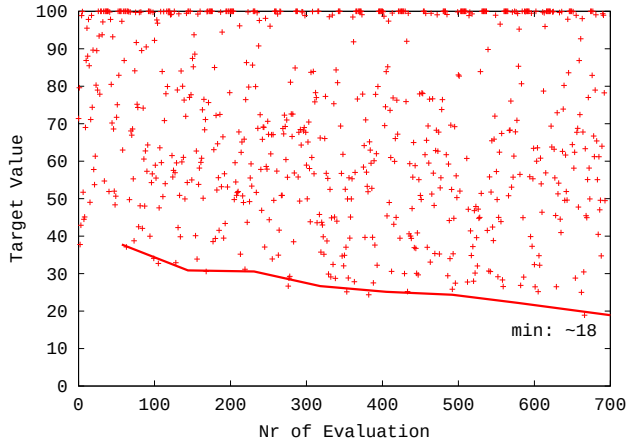

Figure 6: Evolution of the genetic optimizer for $P_{cross} = 0.8$, $P_{mut} = 0.3$ and one-point-crossover

GALIB does not support parallel target evaluation our optimizer takes care of evaluating several jobs in parallel.

The parameters of genopt with the most impact are the crossover probability P_{cross} and the mutation probability P_{mut}. Several experiments with different crossover and mutation probabilities were carried out. Fig. 5 and Fig. 6 depict the evolution of the genetic algorithm for two different combinations of crossover and mutation probability and crossover method. The solid line is a plot of the best individual of each generation. Note that the best individual within a population sometimes occurs at a lower evaluation number thus appearing below the solid line.

The parameter combination depicted in Fig. 5 leads to the best result for our application.

2.3 Simulated Annealing

Simulated Annealing is an optimization technique which was first introduced by Kirkpatrick in 1983 [13]. It is comprised of three functional relationships: The generation function $g(\vec{x})$, where $\vec{x} = \{x^i; i = 1, D\}$ with dimension D, the acceptance function $h(\vec{x})$ and the annealing schedule function $T(k)$ with the time step k. The optimization itself takes place iteratively. Initially, the algorithm starts from a randomly chosen point from which the fitness is computed. Next a new point is chosen using $g(\vec{x})$. In case the fitness of this point is better than the fitness of the other one, the new point is taken over. In case the fitness is worse the point is accepted by a probability $h(\vec{x})$. Another point is always chosen based on the best point so far. With each iteration the probabilities for large deviations from the best point and for acceptance decrease. This results in a behavior where distant points are explored at the beginning (high temperature) but not generated or rejected respectively as the temperature cools down.

For the standard Boltzmann Annealing $g(\vec{x})$, $h(\vec{x})$ and $T(k)$ are given by:

$$g(\vec{x}) = (2\pi T)^{-\frac{D}{2}} \exp\left(-\frac{\Delta\vec{x}^2}{2T}\right), \quad (2)$$

$$h(\vec{x}) = \frac{1}{1 + \exp\left(\frac{E_{k+1} - E_k}{T}\right)}, \quad (3)$$

$$T(k) = \frac{T_0}{\ln k} \quad (4)$$

with the deviation $\Delta\vec{x} = \vec{x} - \vec{x}_0$ of the new state from the previous one. It was shown [14] that a global minimum will be found if the temperature is decreased no faster as given by (4).

Siman

Our simulator (siman) is based on the VERY FAST SIMULATED RE-ANNEALING [3] algorithm by L. Ingber. The algorithm defines a generation rate which allows for an exponentially decreasing time step function:

$$T_i(k) = T_{0i} \exp\left(-c_i k^{\frac{1}{D}}\right) \quad (5)$$

with $c_i = m_i \exp\left(-\frac{n_i}{D}\right)$, where m_i and n_i are tuning parameters. The values T_{0i} are the initial annealing temperatures.

To account for different sensibilities of the parameters the algorithm periodically re-scales the annealing time k. The range over which the more insensitive parameters are searched is stretched out with respect to the more sensitive parameters (RE-ANNEALING).

Fig. 7 shows the progress of siman. Standard parameter settings were used. Compared to genopt this optimizer reaches the same target value within approximately one third of evaluations.

3 CONCLUSION

We conclude that among the global optimization strategies we evaluated, simulated annealing seems to be

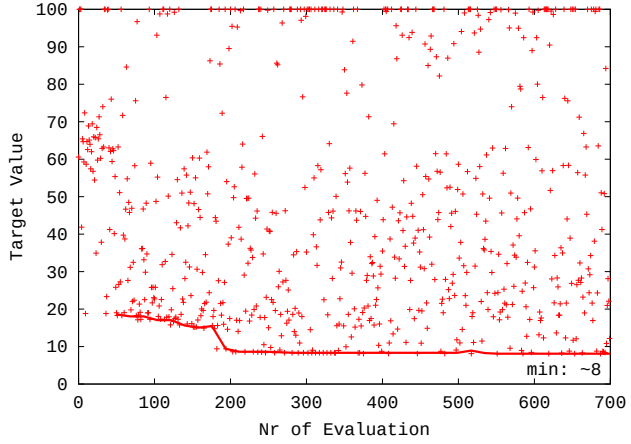

Figure 7: Evolution of the simulated annealing optimizer

best suited for the case of our inverse modeling application. We observed that for a larger number of evaluations (several thousands) siman delivered nearly optimal target values, whereas genopt's optima did not drop below a certain value. This calls for further experimenting with P_{cross} and P_{mut} and other parameters during the evolution. However, the optimal settings for these parameters are difficult to extract. We found that the VERY FAST SIMULATED RE-ANNEALING algorithm is faster than the STEADY-STATE genetic algorithm by at least a factor of three. This conforms to the experiments done by L. Ingber [15] who reports a speed difference of about one magnitude.

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. In this case the whole optimization must be restarted with a different initial guess.

Compared to a local optimizer the presented global optimization techniques demonstrate robust optimization strategies which are essential in cases where an appropriate initial guess is not available.

Further investigations will combine the advantages of global and local optimization techniques. One could imagine a scenario where for each globally found target value below a certain limit (e.g. 15), a separate local optimizer is tried for a certain time period. This combines the robustness of the global technique with the speed of the local one.

ACKNOWLEDGMENT

This work is supported by the "Christian Doppler Forschungsgesellschaft", Vienna, Austria.

REFERENCES

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

[2] M. Wall, "GAlib A C++ Library of Genetic Algorithm Components," *Massachusetts Institute of Technology*, 2000, `http://lancet.mit.edu/ga`.

[3] L. Ingber, "Very Fast Simulated Re-Annealing," *Mathematical Computer Modelling*, vol. 12, pp. 967–973, 1989, `http://www.ingber.com/asa89_vfsr.ps.gz`.

[4] R. Plasun, *Optimization of VLSI Semiconductor Devices*, Dissertation, Technische Universität Wien, 1999, `http://www.iue.tuwien.ac.at/diss/plasun/diss-new/diss.html`.

[5] S. Selberherr, *Analysis and Simulation of Semiconductor Devices*, Springer, Wien, New York, 1984.

[6] R. Strasser, R. Plasun, and S. Selberherr, "Practical inverse modeling with SIESTA," in *Simulation of Semiconductor Processes and Devices*, Kyoto, Japan, Sept. 1999, pp. 91–94.

[7] C. Heitzinger and S. Selberherr, "An Extensible TCAD Optimization Framework Combining Gradient Based and Genetic Optimizers," *in Proc. International Symposium on Microelectronics and Assembly, Singapore 2000*, pp. 279–289, Nov 2000.

[8] T. Grasser, V. Palankovski, G. Schrom, and S. Selberherr, "Hydrodynamic mixed-mode simulation," in *Simulation of Semiconductor Processes and Devices*, K. De Meyer and S. Biesemans, Eds., pp. 247–250. Springer, Leuven, Belgium, Sept. 1998.

[9] 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, *MINIMOS-NT User's Guide*, Institut für Mikroelektronik, 1998.

[10] W.T. Vetterling and S.A. Teukolsky, *Numerical Recipes*, Cambridge University Press, 1986.

[11] J. Holland, "Adaption in Natural and Artificial Systems," *University of Michigan Press, Ann Arbor, MI*, 1975.

[12] D. E. Goldberg, *Genetic Algorithms in Search and Optimization*, Addison-Wesley Pub. Co., 1989.

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

[14] 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, 1984.

[15] L. Ingber, "Genetic Algorithms and Very Fast Simulated Re-Annealing: A Comparision," *Mathematical and Computer Modelling*, vol. 16, pp. 87–100, 1992, `http://www.ingber.com/asa92_saga.ps.gz`.