EPJ Photovoltaics **9**, 13 (2018)

https://doi.org/10.1051/epjpv/2018011

Available online at:
www.epj-pv.org

REGULAR ARTICLE OPEN ACCESS

SLALOM: Open-source, portable, and easy-to-use solar cell optimizer. Application to the design of InGaN solar cells

Sidi Ould Saad Hamady and Nicolas Fressengeas*

Université de Lorraine, CentraleSupélec, LMOPS, 57000 Metz, France

Received: 7 September 2018 / Received in final form: 22 October 2018 / Accepted: 22 November 2018

Abstract. The design and optimization of novel structures is an essential part of the next-generation solar cells development. Indeed, the technological steps involved in the development of high-performance solar cells involve a huge set of interdependent physical and geometrical parameters: layers thicknesses, dopings, compositions, and defect characteristics. In this work, we propose a new open-source and free solar cell optimizer: SLALOM − for SoLAr ceLl multivariate OptiMizer − that implements a rigorous multivariate approach, which improves from the one-parameter-at-a-time procedure that is traditionally used in the field to a state-of-the-art multivariate approach. Applied to indium gallium nitride (InGaN) solar cells, it shows its potential to become a useful tool for the development of novel solar cells. SLALOM is implemented to be extended to any semiconductor simulation engine. Several models for solar cells have been implemented in SLALOM, including, for instance, InGaN. One can adapt these models to any solar cell technology by changing the parameter set, the here proposed generic code structure remaining unchanged.

Keywords: SLALOM / solar cell / optimization / simulation / design / InGaN

Numerical design and optimization are crucial in the development of new solar cell technology, where the main objective is to optimize the solar cell efficiency with respect to the material properties and device technological parameters. This development involves complex steps, including the active layers elaboration, doping, passivation, and device realization, and thus involves a large set of technological and physical parameters to optimize: the layers' composition, their dopings, thicknesses, optical parameters, absorptions, mobilities, and diffusion lengths. All these parameters are interdependent: e.g., the optimal absorption depends on the available layer thickness and the diffusion length, while the optical parameters depend on the composition.

The traditionally used standard experimental or theoretical procedure to study the impact of these parameters, or at least a subset of them, is the *parametric analysis*: the solar cell efficiency variation is studied with respect to the variation of one arbitrarily chosen parameter, while the other parameters are kept constant [1–5]. One experimental example of this procedure could be the study of the impact of the N-doping on the solar cell efficiency by elaborating a solar cell with multiple N-doping concentrations in a feasible range, while keeping the P-doping, the composition, the layers' thicknesses, and all the other parameters constant. Similarly, the theoretical procedure would study by numerical simulation the impact of the N-doping concentration on the solar cell efficiency while keeping all the other parameters constant. The experimental and the theoretical procedures suffer from two killing drawbacks. First, and whatever the choice of the optimized parameters, the method does not converge toward the optimum efficiency as it ignores parameters' interdependence. In our example, the *optimal* N-doping is only optimal for the chosen values for the P-doping and the other parameters such as the absorption and the diffusion length. The second drawback of the *parametric analysis* is the fact that it requires a huge number of evaluations of the efficiency, so large that their effective evaluation is hardly possible. For instance, the simultaneous impact of N-doping, P-doping, N-layer, and P-layer thicknesses could be studied by using 400 samples (100 for each parameter variation range) that are experimentally quasi-impossible and theoretically very long to perform. Indeed, each complete simulation usually takes up to 10 min on a high-performance calculation server, leading to days of computing for the whole problem.

* e-mail: nicolas.fressengeas@univ-lorraine.fr

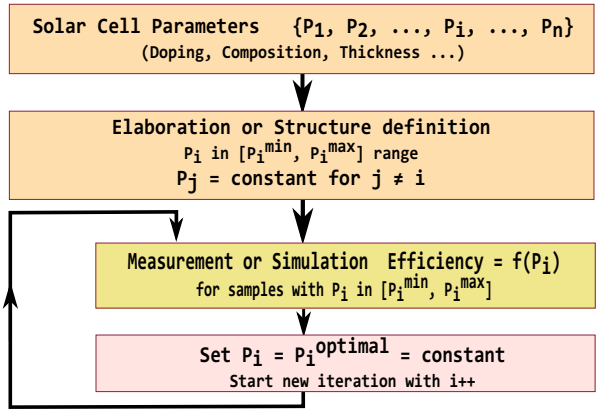

Fig. 1. The parametric analysis procedure.

The method that is proposed in this paper brings a solution to both drawbacks by using a multivariate approach implemented in a complete free and open-source software: SLALOM, which stands for "SoLAr ceLl multivariate OptiMizer."[1] This approach is shown to be altogether effective, precise and efficient in time, thus drastically decreasing the optimization time and consequently the overall solar cell development time.

In the next section the mathematical procedure used in SLALOM is detailed. In Section 2, the software implementation is reviewed thoroughly. The SLALOM method and software are then applied to InGaN-based solar cells and the results are discussed in Section 3, before a final conclusion.

1 Solar cell multivariate optimization

1.1 Review of the traditional parametric analysis

The parametric analysis described in the previous section is illustrated in Figure 1.

The first step in this standard one-by-one parametric analysis consists in defining a set of technological and physical parameters: for instance, for a compound superconductor, its composition, its P- and N-type doping concentrations, its layers' thicknesses, among other equally important parameters.

Depending on whether the undertook optimization is theoretical or experimental, the second step consists in defining the structure set to simulate or actually elaborating the samples within that set. The set is made up of n samples with only one varying parameter. For instance, one could elaborate 10 samples, or simulate them, with a composition x varying from 0.1 to 0.5 while the P- and N-type doping concentrations, layers' thicknesses, and all other parameters are kept constant among all the samples.

[1]SLALOM source code is available for download on GitHub (https://github.com/sidihamady/SLALOM), HAL (https://hal.univ-lorraine.fr/hal-01897934) and on the author website (http://www.hamady.org/photovoltaics/slalom_source.zip). Further instructions for its effective use are given in Appendix A.

The third step consists in characterizing or simulating these samples to extract the main performance figure, that is, for solar cells, the photovoltaic efficiency. The sample for which the efficiency is optimal in then identified and the varying parameter set to the optimal value. The procedure then iterates for the next parameter, keeping all other parameters constant.

This procedure, albeit simple to handle, suffers from two major drawbacks. The first one is its mathematical uncertainty: at the end of the procedure, the highest obtained efficiency is not guaranteed to be the optimal one since the procedure ignores the interdependence between the solar cell parameters. For instance, an optimal composition obtained with this procedure is only valid for the set of doping concentrations and layers' thicknesses that were kept constant in the set. If the procedure had been started with a different parameter set, the optimal composition found would have been different. The direct practical consequence of this issue is that the maximum feasible efficiency is actually missed. An indirect consequence is that the actual optimum could corresponds to a set of parameters that is easier to achieve practically than the found one.

The second major drawback of this parametric analysis is its time-consumption. If performed experimentally, it implies the elaboration of a high number of samples, increasing the overall cost. Besides, the elaboration of a set of solar cells with one and only one varying parameter is not straightforward. Usually, these difficulties are addressed by elaborating a smaller set of samples based on the user experience in the material and process. This procedure is however not usable when developing a new technology. On the simulation side, this drawback is only related to the calculation time. Usually, with standard industry simulators, a single complete solar cell simulation, outputting current–voltage characteristic under solar AM1.5 standard illumination as defined at NREL [6], takes a few minutes on a server with two 8-core Xeon processors and 32 GB of RAM. It can however take up to 20 min and more if more complex structure and more complete models are considered. In this one-by-one parametric procedure, the total needed time varies linearly with the number of parameters, both for experimental measurement and theoretical determination of the optimal parameters.

1.2 Review of the brute force optimization

A second procedure sometimes used is the so-called *brute force* optimization. In this procedure, each parameter varies in a given range. The procedure iterates, on each value of a given parameter, in the whole n-dimension space – n being the parameter number – so that *all possible combinations are tested*. The total number of characterizations or simulations needed is thus equal to m^n, where m is the number of points taken within each parameter range.

For instance, if one deals with 5 parameters and 10 samples per parameter, the total number of characterizations or simulations will be equal to 10^5. Even for this small number of samples, this method is absolutely not usable for solar cells, since the individual simulation time

are on the order of tens of minutes, implying years for a single optimization on our server with two 8-core Xeon processors and 32 GB of RAM.

The brute force optimization is therefore completely unusable experimentally, or even numerically, for more than two or three parameters. Its only advantage is that it guaranteed to yield the absolute optimum within the tested points. This advantage is however completely annihilated by the fact that an acceptable precision can only be obtained for a high number of points within each parameter range, implying again a crippling computation time.

1.3 Newton-based optimization methods

The third and last procedure, the one that is proposed in this paper, consists in using state-of-the-art mathematical optimization algorithms to find the parameter set that yields the optimal solar cell efficiency. These algorithms take the parameters interdependency into account. They have been carefully designed to use the least possible computing time, leading to a very small need in computer resources if compared to the previously described basic procedures. This mathematically secured procedure is largely used in physical engineering areas such as mechanical, electrical, or civil engineering [7–9]. We have used this method for the design and the optimization of solar cells [10–12] and propose with this paper, to fully detail and completely release it and the software behind it for the free use of all, under the *MIT License*.

Given a set of parameters pertaining to the solar cell operation (e.g., composition, doping concentrations, layers thicknesses), it goes by expressing the overall solar cell efficiency η as a nonlinear function f, called *the objective function*, of the parameter set expressed as an n-dimension vector $\vec{P}$:

$$\eta = f(P_1, P_2, P_3, ..., P_n) = f\left(\vec{P}\right). \quad (1)$$

This function f can be the mathematical form of any procedure, whether it be an algebraic expression, the result of a numerical computation, or even that of experimentations. It takes as its input the given set of parameters and yields the value of the corresponding solar cell efficiency. In the case presented in this paper, the evaluation of the objective function f requires the use of a semiconductor simulation engine that allows the numerical evaluation of the solar cell efficiency from its known physical and geometrical parameters.

Once f is carefully defined, one can use it inside a mathematical algorithm designed to find the parameter vector that maximizes the efficiency f using an initial guess and a numerical iterative procedure. Such algorithms are numerous in the literature and are available in freely usable software libraries such as SciPy [13].

We mainly used two of these algorithms that we demonstrated, after a comprehensive study involving 10 of them, to be suitable for the optimization of solar cells.

The first one is the Sequential Least Squares Programming (SLSQP) method [14]. The iterative algorithm used in it starts from an initial guess, a given vector of parameters $\vec{P}^{(k)}$, and constructs the next vector $\vec{P}^{(k+1)}$ by using Newton's method to find the best direction, where the gradient function vanishes, in the n-dimensional parameters space. After a number of iterations, the SLSQP method converges to the point maximizing the objective function, i.e., the solar cell efficiency in our case. Each parameter in the objective function is constrained: it is allowed to vary in a range defined by the technological feasibility and the underlying physics. An alternative, the SQP method [14] can handle more complex constraints including nonlinear equality and inequality constraints.

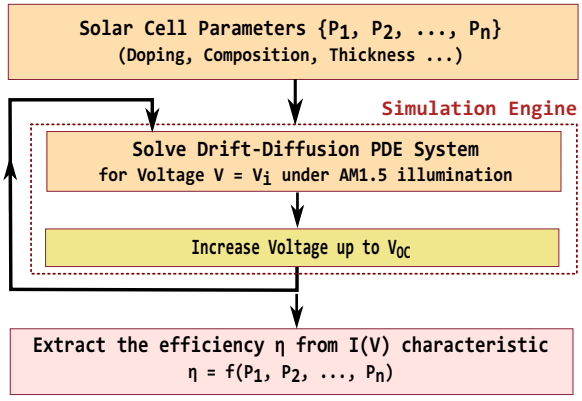

Fig. 2. The procedure used for extracting the solar cell efficiency by solving the coupled Poisson and continuity PDE system for voltage ranging from 0 to V_{OC}, the open-circuit voltage.

The second algorithm that drew our interest is the Broyden–Fletcher–Goldfarb–Shanno (BFGS)-based iterative algorithm [15]. It uses a quasi-Newton method seeking for the maximum of a nonlinear function of n variables using gradient evaluations and approaching the second derivatives (Hessian matrix). The BFGS algorithm (i) starts from a given initial vector $\vec{P}^{(0)}$ and approximated Hessian matrix; (ii) then it calculates the best direction in the n-dimensional parameters space using the Hessian matrix; (iii) next it updates the solution vector using a step size in the found direction; (iv) thereafter it updates the Hessian matrix using the new vector; and finally it iterates all these steps until the obtained solution is bracketed within the desired precision. This method has the same convergence criteria as the Newton method while it is demonstrated to be faster than the other quasi-Newton methods. Two versions of this algorithm are used in this work: the L-BFGS, L for limited memory that better handles a high number of parameters, and, more crucial for our application, the BFGS-B algorithm (B for bound constrained) that is the constrained version. The L-BFGS-B method has the criteria needed in our application: performance and constraints.

As hinted earlier, the function f that we need to optimize could be any nonlinear explicit function, or an experimental procedure, or extracted from a solution of coupled partial differential equations (PDE). In our case, the function f is described in Figure 2: the Poisson and continuity PDE system is solved by a semiconductor device simulator under solar AM1.5 [6] illumination in a large voltage range. The corresponding current is calculated for each voltage value.

The photovoltaic efficiency is then extracted from this simulated current–voltage characteristic.

The mathematical algorithm takes this function f and gives, after a number of iterations, the parameter vector maximizing the efficiency. The two suitable optimization algorithms presented in the present section, SLSQP and BFGS, were used and throughly tested with respect to two major criteria: the number of function evaluations (i.e., the overall optimization speed) and its robustness (i.e., convergence in a wide range of initial parameter values).

All this was implemented in the SLALOM open-source piece of software, using the open-source SciPy library [13] for optimization and opening for various semiconductor simulator engines. The next section describes in detail this implementation in order to provide the reader with the ability to use the SLALOM software for her or his own purpose.

2 The multivariate optimizer implementation

2.1 SLALOM architecture

Figure 3 shows the SLALOM general architecture. The basic rules used in the implementation of SLALOM were as follows:

- Simplicity – KISS principle: *Keep It Simple and Straightforward*;
- Modularity and modification friendliness;
- Code correctness and efficiency.

SLALOM is written in vanilla Python [16] but it could be re-implemented in other interpreted languages (such as Lua) or compiled ones (such as C++). It uses the SciPy [13] and NumPy [17] numerical packages, which are Python interfaces to a collection of state-of-the-art numerical and scientific routines written in C and Fortran. These routines are freely available from http://netlib.org and could be accessed by any language that can interface C routines. The monitoring and visualization part of SLALOM was written using Tkinter [18] for the user interface and Matplotlib [19] for 2D curve plotting.

It is composed of seven main modules:

- slalom.py, the SLALOM startup module, sets the parameters, the optimization method and controls the whole process.
- slalomCore.py, the SLALOM core class, implements the optimization using one of the mathematical algorithms provided by the SciPy package, and controls the device simulation engine. As a class with defined and modularized functionalities, it can be extended by creating a new inherited class. This inheritance mechanism allows modularity, clearness, and reliability by keeping unchanged a highly tested base code.
- slalomDevice.py is a class defining a set of devices (e.g., InGaN_PN) for easier and more robust optimization work. For any project, this class can be reimplemented to include any set of relevant devices.
- slalomDeviceGui.py is a class providing an interface to create a new device type. It is only used if the device type specified in slalom.py is not defined in slalomDevice.py. slalomDeviceGui.py uses the Tkinter graphical toolkit that is assumed already to be installed on the client (this is generally the case, except for some CentOS or Red Hat machines). If Tkinter is not installed, this class is not used.
- slalomMonitor.py is used to monitor the optimizer either in client–server configuration using SSH or locally if it runs on the same machine as the optimizer. slalomMonitor uses the slalomWindow class that provides visualization and control functionalities. If Tkinter is not installed, this class is not used.
- slalomSimulator.py is the class interfacing the solar cell simulator engine. This class encapsulates to a common python interface the functionalities that are specific to the simulator, either commercial simulators such as Silvaco® and tiberCAD® or free simulators such as Afors-HET [20], SCAPS [21], PC1D [22], and AMPS-1D [23]. It can furthermore be extended to include any simulator that can be launched from the command line and that outputs its results in text files (i.e., any well-designed simulator).
- slalomWindow.py: it the common class providing the visualization and control functionalities used by the SLALOM Graphical User Interface part. It is only available if Tkinter is installed.

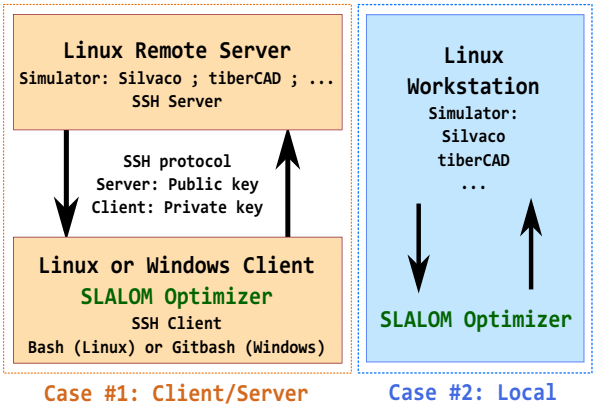

Fig. 3. The SLALOM optimizer general architecture.

2.2 Customizing SLALOM

SLALOM is freely available under the MIT license and continuously updated. As for now, SLALOM natively supports the widely used Silvaco® Deckbuild/Atlas simulation engine and tiberCAD®, and is being extended to support free simulators such as Afors-HET [20], SCAPS [21], PC1D [22], and AMPS-1D [23]. Extending it to yet another solar cell simulator is straightforward if it is designed in the command-line scriptable standard way, which is the case for all well-designed simulators: command line interface, well-defined syntax in input raw text files including device parameters, output to raw text files.

For the Silvaco® tools, the optimizer constructs the device input in a modular way: the Deckbuild input file includes C files that define the physical models for the

Table 1. Bandgap E_g, affinity X, permittivity ε, and densities of states in the conduction band N_C and in the valence band N_V of GaN and InN [29–31]. The absorption is given by equation (2), the C and D values being fitting from experimental data [32], with E_{ph} the incident photons energy and E_g the InGaN bandgap for a given indium composition x.

	E_g(eV)	X(eV)	N_C(cm^{-3})	N_V(cm^{-3})	ε
GaN	3.42	4.1	2.3×10^{18}	4.6×10^{19}	8.9
InN	0.7	5.6	9.1×10^{17}	5.3×10^{19}	15.3

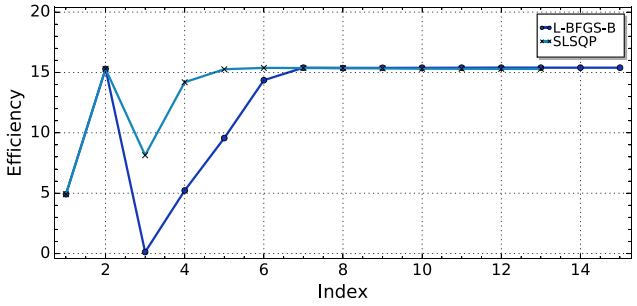

Fig. 4. Efficiency of the InGaN PN solar cell determined by SLALOM using the L-BFGS-B and SLSQP methods, with the initial point chosen relatively far from the global optimum.

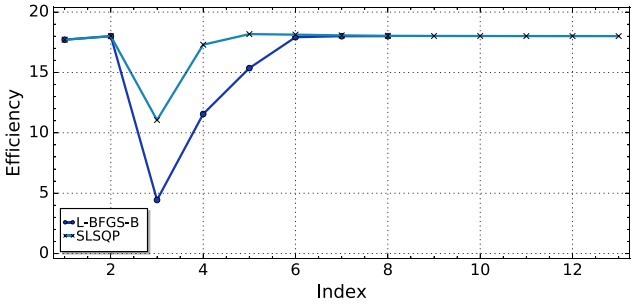

Fig. 5. Efficiency of the InGaN PN solar cell determined by SLALOM using the L-BFGS-B and SLSQP methods, with the initial point choosen in the close vicinity of the global optimum.

bandgap, the mobility, the refractive index (both for its real and imaginary parts that models absorption), and the recombination mechanisms along their dependence on composition, temperature.

This architecture offers the ability to control in detail the physical models and to change the parameters in a set of small and consistent files. All these files share one C header file including the main material parameters. These files are used by the optimizer to create the simulation input set that it uploads to the calculation server if needed. The optimizer then controls Deckbuild and monitors its output.

For the tiberCAD® tool, the same methodology as for Silvaco® is used, and could have been used for any other command line scriptable simulator. The software has only been adapted to the tiberCAD® syntax constraints: each optimization parameter is recognized with a specific comment line before the corresponding statement.

3 Application to InGaN solar cells

This next section presents the main results of the application of SLALOM to InGaN PN junction solar cells and details about the software behavior for a higher parameter number.

3.1 Optimization of an InGaN PN junction solar cell

The developement of high-efficiency and low-cost thin films third-generation solar cells are crucial for the coming next 30 years. The InGaN alloy has the potentiality to play a major role in their development, thanks to its high absorption coefficient [24] and resistance to extreme conditions such as high temperature or high irradiance [25] and thanks to its bandgap that can be tuned and adapted to almost the whole solar spectrum [26]. Nevertheless, this potential is not yet exploited and the best photovoltaic efficiency reported today does not exceed 3% [27,28]. This efficiency is indeed still highly limited by the not-high-enough quality of the elaborated layer for high indium composition, along with the difficulties to realize ohmic contacts and to grow p-doped InGaN.

In this context, we used SLALOM to precisely seek the optimal efficiency that can be achieved by InGaN PN junction solar cells, using as-realistic-as-possible physical models with parameters extracted from experimental data, with the aim to simultaneously optimize the cell 5 following parameters : the Indium composition alongside the P and N layers dopings and thicknesses. These parameters and the used models have been detailed previously [10,11]. Table 1 and equation (2) give the values that were taken for the bandgap, the affinity, the density of states, and the absorption. To ease its use with this particular configuration, the complete simulation files for Silvaco® are packaged with the SLALOM distribution.

$$\begin{aligned}
\alpha(\mathrm{cm}^{-1}) &= 10^5 \sqrt{C\left(E_{\mathrm{ph}} - E_g(x)\right) + D\left(E_{\mathrm{ph}} - E_g(x)\right)^2} \\
C\,(\mathrm{eV}^{-1}) &= 3.525 - 18.29\,x + 40.22\,x^2 - 37.52\,x^3 + 12.77\,x^4 \\
D\,(\mathrm{eV}^{-2}) &= -0.6651 + 3.616\,x - 2.460\,x^2
\end{aligned} \tag{2}$$

Figures 4 and 5 show the efficiency of the InGaN PN solar cell as determined by the optimizer using the L-BFGS-B and SLSQP methods with the initial point choosen far from the global optimum (first case, Fig. 4) and near it (second case, Fig. 5). Each index corresponds to a set of parameters choosen by the optimizer in the variation domain defined in slalomDevice.py. Both methods converge to the same set of optimal parameters

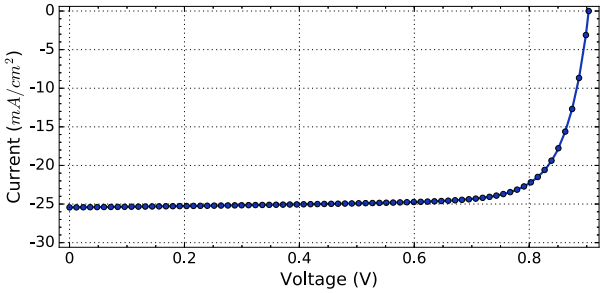

Fig. 6. Current–voltage characteristic of the InGaN PN solar cell at the optimum.

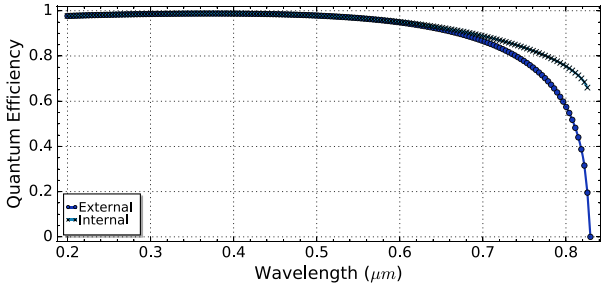

Fig. 7. External and internal quantum efficiency spectra of the InGaN PN solar cell at the optimum.

in about 2 h for L-BFGS-B and 1 h for SLSQP on a Red Hat Linux server with two 8-core Xeon processors and 32 GB of RAM. When the initial point is choosen in the close vicinity of the global optimum (second case, Fig. 5), the algorithm converges faster to the optimum (50 min for SLSQP and 90 min for L-BFGS-B). This point illustrates the fact that it is necessary to use a *reasonable* strategy to efficiently find the global solar cell optimal efficiency considering the complexity of the underlying physical model (drift-diffusion) and its nonconvex nature.

The optimal set of parameters gives an optimal efficiency of about 18%, a short-circuit current of 26.8 mA/cm^2, an open-circuit voltage of 0.85 V, and a fill factor of 78%. It lies in the vicinity of the following set of values: a P-layer thickness of 0.01 µm, an N-layer thickness of 1 µm, a P-layer doping of 10^{19} cm^{-3}, an N-layer doping of 4×10^{16} cm^{-3}, and an indium composition of 56%. Figure 6 shows the current–voltage characteristic near the optimal point, while Figure 7 plots the external and internal quantum efficiency spectra.

Unfortunately, this very high efficiency is not reachable for many technological reasons, among which, for instance, the required 56% indium composition, which is not feasible today. Solving this problem requires optimizing the solar cell while taking into account the limits of material technology, mainly related to InGaN elaboration and to solar cell fabrication. This can be done in SLALOM by setting the optimization parameters range to technology limits in slalomDevice.py: the corresponding optimizations and results can be found in previously published literature [11,12].

For comparison purpose, we can evaluate the time that would have been needed for the same optimization if we had used the brute force method, with only five samples per parameter: the total number of points would have been $5^5 = 3125$. The total duration would have been about 100 h on the same machine, as we have tested. The parameter precision would have been equal to the range divided by the number of samples per parameter. For thickness, for example, the resolution would have been about 0.2 µm (for five points per parameter), meaning that the optimal thickness is known within this uncertainty. Seeking higher resolution renders the brute force method become completely infeasible: its execution time varies as n^m, where n is the number of points and m the number of parameters. Again, as a comparison, for the optimization methods (L-BFGS-B and SLSQP) the resolution is determined by the Jacobian step, which is, in this case, equal to the range divided by 50, meaning 0.02 µm, 10 times better than previously. Therefore, with a far better resolution, the L-BFGS-B and SLSQP methods are *at least* two orders of magnitude faster than the brute force method.

3.2 Optimization with many parameters

For the sake of clarity, this paper only presents solar cell optimizations with a reduced number of parameters, namely, five for the InGaN PN junction. Detailed applications of SLALOM with up to 11 parameters can be found in references [10–12].

Using SLALOM with up to 11 parameters, and probably more, does not raise specific adaptation or coding difficulties, as the underlying mathematical method has been shown to work for any parameter number. However, specific convergence issues can arise from the finite precision inherent to any computer calculation. Indeed, a rapid and precise convergence for a high number of parameters requires an all the more precise evaluation of the function to be optimized.

This computer-specific limitation leads to a limit, which is above 11 on our Xeon processors, of the number of parameters that can be optimized simultaneously. This limit can however be overcome by carefully choosing the optimization starting point close to the expected optimum. This starting point can itself be found using approximate physics arguments, or an optimization with a limited number of parameters, or both.

4 Conclusion

A new solar cell optimization method is presented in this paper. This method is implemented in SLALOM, an easy-to-use, portable, and open-source software. SLALOM implements mathematical multivariate methods for the rigorous optimization of solar cell parameters taking into account the parameters' interdependencies and speeding up the optimization procedure by two orders of magnitude. SLALOM is freely distributed with a complete set of cases implementing physical models and data extracted from experimental measurements for realistic and precise solar cell simulation. Its use was presented in this paper for the optimization of an InGaN PN solar cell. The absolute optimal efficiency was obtained for a set of five parameters crucial to the solar cell design and elaboration: the indium composition, thicknesses, and doping concentrations for the InGaN structure.

Author contribution statement

The authors contributed equally to the paper, the former focusing on semiconductor physics and actual coding and the latter on the optimization procedures.

Appendix A: SLALOM guide for free download and use

The SLALOM piece of software is distributed under an open-source MIT license for free use, modification, and redistribution. Its source code is available for download and use from three sources:

- the international GitHub platform, at https://github.com/sidihamady/SLALOM,
- the French Open Archive initiative HAL, at https://hal.univ-lorraine.fr/hal-01897934,
- the author website, at http://www.hamady.org/photovoltaics/slalom_source.zip.

A comprehensive installation and use guide is included in the distribution and can be downloaded here : https://github.com/sidihamady/SLALOM/blob/master/Guide/slalom_guide.pdf.

Finally, the authors will naturally answer happily to any solicitation regarding help in installing or using SLALOM.

References

1. X. Zhang, X. Wang, H. Xiao, C. Yang, J. Ran, C. Wang, Q. Hou, J. Li, J. Phys. D: Appl. Phys. **40**, 7335 (2007)
2. S.W. Feng, C.M. Lai, C.H. Chen, W.C. Sun, L.W. Tu, J. Appl. Phys. **108**, 093118 (2010)
3. H. Movla, D. Salami, S.V. Sadreddini, Appl. Phys. A **109**, 497 (2012)
4. M. Nawaz, A. Ahmad, Semicond. Sci. Technol. **27**, 035019 (2012)
5. J.Y. Chang, Y.K. Kuo, IEEE Electron Device Lett. **32**, 937 (2011)
6. NREL, Reference Solar Spectral Irradiance: ASTM G-173 (2004)
7. A.K. Hartmann, H. Rieger, in *Optimization algorithms in physics* (CiteSeer, 2002), Vol. 2
8. K.S. Lee, Z.W. Geem, Comput. Methods Appl. Mech. Eng. **194**, 3902 (2005)
9. E.S. Mistakidis, G.E. Stavroulakis, in *Nonconvex optimization in mechanics: algorithms, heuristics and engineering applications by the FEM* (Springer Science+Business Media, Berlin, 2013), Vol. 21
10. S. Ould Saad Hamady, A. Adaine, N. Fressengeas, Mater. Sci. Semicond. Process. **41**, 219 (2016)
11. A. Adaine, S. Ould Saad Hamady, N. Fressengeas, Superlattices Microstruct. **96**, 121 (2016)
12. A. Adaine, S. Ould Saad Hamady, N. Fressengeas, Superlattices Microstruct. **107**, 267 (2017)
13. E. Jones, T. Oliphant, P. Peterson, et al., *SciPy: Open source scientific tools for Python*, http://www.scipy.org/ (2001)
14. S.J. Wright, J. Nocedal, *Numerical optimization* (Springer, Berlin, 1999)
15. R.H. Byrd, P. Lu, J. Nocedal, C. Zhu, SIAM J. Sci. Comput. **16**, 1190 (1995)
16. G. van Rossum, *Python programming language*, https://www.python.org/ (2017)
17. S. van der Walt, S.C. Colbert, G. Varoquaux, Comput. Sci. Eng. **13**, 22 (2011)
18. J.W. Shipman, *Tkinter 8.5 reference: a GUI for Python*, www.nmt.edu/tcc/help/pubs/tkinter/tkinter.pdf (2013)
19. J.D. Hunter, Comput. Sci. Eng. **9**, 90 (2007)
20. R. Stangl, M. Kriegel, M. Schmidt, Afors-het, version 2.2, a numerical computer program for simulation of heterojunction solar cells and measurements, in *2006 IEEE 4th World Conference on Photovoltaic Energy Conversion* (IEEE, 2006), Vol. 2, pp. 1350–1353
21. M. Burgelman, P. Nollet, S. Degrave, Thin Solid Films **361**, 527 (2000)
22. P.A. Basore, D.A. Clugston, PC1D version 4 for Windows: from analysis to design, in *Conference Record of the Twenty Fifth IEEE Photovoltaic Specialists Conference, 1996* (IEEE, 1996), pp. 377–381
23. H. Zhu, A.K. Kalkan, J. Hou, S.J. Fonash, Applications of AMPS-1D for solar cell simulation, in *AIP Conference Proceedings* (AIP, 1999), Vol. 462, pp. 309–314
24. E. Matioli, C. Neufeld, M. Iza, S.C. Cruz, A.A. Al-Heji, X. Chen, R.M. Farrell, S. Keller, S. DenBaars, U. Mishra, Appl. Phys. Lett. **98**, 021102 (2011)
25. S.J. Pearton, F. Ren, E. Patrick, M.E. Law, A.Y. Polyakov, ECS J. Solid State Sci. Technol. **5**, Q35 (2016)
26. P.G. Moses, C.G. van de Walle, Appl. Phys. Lett. **96**, 021908 (2010)
27. C.A.M. Fabien, A. Maros, C.B. Honsberg, W.A. Doolittle, IEEE J. Photovolt. **6**, 460 (2016)
28. N.G. Young, E.E. Perl, R.M. Farrell, M. Iza, S. Keller, J.E. Bowers, S. Nakamura, S.P. DenBaars, J.S. Speck, Appl. Phys. Lett. **104**, 163902 (2014)
29. V.Y. Davydov, A.A. Klochikhin, V.V. Emtsev, S.V. Ivanov, V.V. Vekshin, F. Bechstedt, J. Furthmüller, H. Harima, A. V. Mudryi, A. Hashimoto, Phys. Status Solidi B **230**, R4 (2002)
30. V.M. Polyakov, F. Schwierz. Appl. Phys. Lett. **88**, 032101 (2006)
31. J. Wu, W. Walukiewicz, Superlattices Microstruct. **34**, 63 (2003)
32. G.F. Brown, J.W. Ager, III., W. Walukiewicz, J. Wu, Sol. Energy Mater. Sol. Cells **94**, 478 (2010)

Cite this article as: S. Ould Saad Hamady, N. Fressengeas, SLALOM: Open-source, portable, and easy-to-use solar cell optimizer. Application to the Design of InGaN Solar Cells, EPJ Photovoltaics 9, 13 (2018)