International Journal of Advanced Engineering Research and Science (IJAERS)
ISSN: 2349-6495(P) | 2456-1908(O)
Vol-8, Issue-1; Jan, 2021
Journal Home Page Available: https://ijaers.com/
Journal DOI: 10.22161/ijaers
Article DOI: https://dx.doi.org/10.22161/ijaers.81.35

Calibration of the SLEUTH urban simulation model using NOMAD and Genetic Algorithms

André Koscianski, Leonardo Pedrozo Amaral

Department of Informatics, Universidade Tecnológica Federal do Paraná (UTFPR), Ponta Grossa, Brasil

Received: 21 Nov 2020;
Received in revised form:
14 Jan 2021;
Accepted: 20 Jan 2021;
Available online: 30 Jan 2021

Keywords— ***Optimization, NOMAD, Genetic Algorithms, Sleuth, Urban Simulation.***

Abstract— *Computer simulations often entail an optimization problem, corresponding to the calibration of model parameters to ensure a precise representation of a given scenario. Many complex phenomena such as urban growth have characteristics that make optimization harder; this can be exemplified by the lack of an analytical formulation, presence of nonlinearity, discontinuities, and nondeterminism. SLEUTH is a long-established urban simulator, used to compute forecasts of city evolution. The tool is controlled by five parameters that span a search space of the order of 10 billion combinations, with a calibration procedure that is CPU-intensive and not compatible with gradient-descent methods. In this work we compare the efficiency of a genetic-algorithm version of the simulator with the use of the optimization library NOMAD. Different alternatives for the integration of the library are suggested. The experiments are analyzed using data profiles, a technique designed to handle cases with a limited number of function evaluations. The results confirm interest in NOMAD, and reveal more information than traditional comparisons of number of iterations or final optimization results. The methodology of the study can be applied to similar situations and is not restricted to simulators implementing urban models.*

I. INTRODUCTION

Simulation is a fundamental tool in science and technology, in fields as different as medicine, astronomy, economics, and engineering. The execution of simulation models makes it possible to compare alternative explanations to a phenomenon, make projections, and refine designs in contexts as research and development of new technological products, or administration of complex systems. Urban simulation is a comparatively recent field, and it encompasses several aspects of the study of cities, such as mobility, waste management, microclimate and urban growth.

A critical point of simulation studies is to guarantee a match between computed results and actual data [1-3]. In other terms, given a physical system, a model should be capable of reproducing its behavior within a tolerance. This can be assessed by monitoring variables of the model and comparing them against reference values, examining qualitative or behavioral characteristics of the simulator, or possibly, conducting the analysis of both types of information [4,5]. In a broad context, finding the correct values for physical parameters that match a phenomenon is generally known as an inverse problem [6]. Computer software may also require calibration of variables that do not directly represent physical properties, but control simulation behavior; this is often the case with complex systems [5].

Models of urban growth, or more generally, models of land use and cover change, require Geographical Information Systems data, which are generally represented as tables of values or 2D images [5]. A common technique to implement urban simulation is Cellular Automata [7,8], where each cell represents a square area of land. Calibration of such models is based on global statistics

(e.g., urbanized area) and local information (e.g., morphology of the urban footprint) [7].

The SLEUTH simulator is a long-established model of urban growth [9,10]. The calibration procedure adjusts the values of five parameters that control the simulation. The parameters take integer values between 0 and 100, meaning that the search space has 101^5 combinations. Calibrating SLEUTH corresponds to finding the extreme of a metric. The original method is based on a brute-force strategy, conducted manually; processing times of the order of six months were reported ten years ago [10]. The software has support for parallel execution via MPI (Message Passing Interface), but this translates to pushing further to the hardware the inefficiency of the search. One attempt to address the problem was the implementation of a genetic algorithm, tailored to this software [10,11]. Studies of model calibration can be found in many fields; cellular automata models of urban growth constitutes part of the literature of the problem, and some references are [12-16].

It has been documented that the SLEUTH model exhibits high sensitivity to temporal locality of data [17]; in our experience, the opposite happens with its control parameters. By inspecting the output of metrics it can be observed the presence of a pattern of plateaus. Other characteristics of the model, as strong nonlinearity, and discontinuity, prompt the use of zero-order optimization methods [18].

In this work, we modify the SLEUTH simulator by coupling it with the optimization toolkit NOMAD [19], capable of handling characteristics as non-convexity and noise. The software connection was implemented using named sockets. The technique of data profiles was used to analyze the results. The performance of NOMAD was compared with the original method of the simulator and also with another version of the tool, GA-SLEUTH, which implements a genetic algorithm for calibration.

II. MATERIALS AND METHODS

2.1 THE SLEUTH MODEL

The software SLEUTH implements a model of urban growth based on the technique of Cellular Automata [8-10]. Each cell obeys a set of rules that determines whether a location becomes urbanized or not, depending on the state of neighbor cells and on five layers of data that give the software its name: Slope, Land Use, Excluded Areas, Urbanization, and Hillshade. The model implements heuristic and stochastic rules that are controlled by five parameters that vary in the range $0 \leq p \leq 100$; they are:

- diffusion: controls the generation of new cells scattered on the matrix;
- bread: controls the expansion of new city fragments;
- spread: controls the generation of new cells around areas that are already urbanized;
- slope: controls to which extent the city can advance over steep terrains; and
- road gravity: controls the generation of new urban cells along of roads.

These five parameters are dubbed ‘SLEUTH DNA’; they are proposed as a means to characterize the dynamics of a city according to the model [20].

Before calculating a forecast, SLEUTH must be calibrated to replicate the historical evolution of a city. This task employs images depicting the past of the area, and a comparison metric. The original procedure is based on an exhaustive search, following the logic on Fig. 1. The simulator is executed with images of increasing finer resolution and smaller parameter grids, producing logs of statistics that are manually inspected [9].

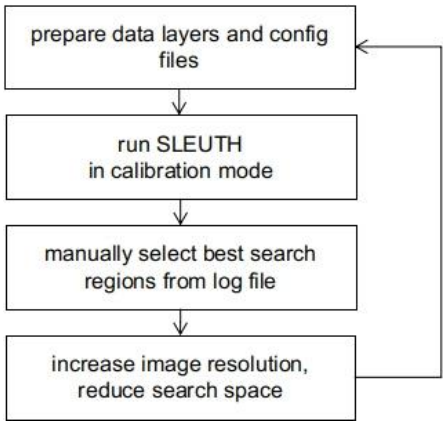

Fig. 1: Original SLEUTH calibration process.

The brute force method scans the search space using five nested loops, with ranges configured by the user. The documentation suggests three calibration rounds, each time doubling image resolution. Each simulation run iterates an internal Monte-Carlo process to average random effects implemented in the model [9,10,21].

The algorithm in Fig. 1 explores contiguous regions and, in principle, does not handle disjoint subsets of parameters. This incurs the risk of losing an extreme point, possibly a global one. To the best of our knowledge, this aspect seems to be overlooked by the literature.

Fig. 2 shows a typical output of one of SLEUTH statistics, obtained in our experiments.

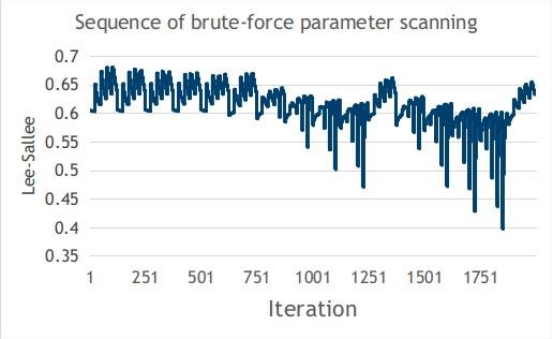

Fig. 2: Output pattern of SLEUTH calibration.

Fig. 2 shows a plot of the metric Lee-Sallee [22], which measures the match between a reference map of urban/non-urban pixels, and the simulator output. Another metric consists of a product of several statistics and is known as OSM (Optimal SLEUTH Metric) [23]. It exhibits much more noise but also has a pattern-like structure. This metric is hardcoded in the genetic version, GA-SLEUTH [10,11]. Both metrics have values between 0 (worst) and 1 (best).

2.2 OPTIMIZATION TOOLS

Direct search methods stand out for their capability to perform optimization without requiring derivatives, and handling nonconvexity and discontinuities [18]. Since such methods make little or no assumptions regarding the behavior of functions, they are also known as black-box optimization algorithms [24,25].

Possibly one of the most famous direct search methods is the one proposed by Nelder and Mead in 1965, which scans the search space using a simplex [26]. A simplex is a polytope with n+1 vertices, where n is the dimension of the search space. Since its introduction, this algorithm has been studied and recast in different forms, including methods to handle constraints and discrete grids [25,27].

NOMAD is a library written in C++ that implements black-box optimization algorithms. It is capable of handling discontinuities, constrained optimization, and functions of discrete variables [28-30]. The algorithm it implements can be divided into two parts [29]:

- search: evaluates *f* for a set of points that radiate to directions D;
- poll: if the search step fails to improve the function, the grid size is adjusted and a different set of directions D' is used to generate candidates.

The implementation allows to modify the default values of parameters and even to change aspects of execution; for instance, the Nelder-Mead algorithm can be chosen as the search step [29]. There are two basic ways to use the NOMAD toolkit, illustrated in Fig. 3.

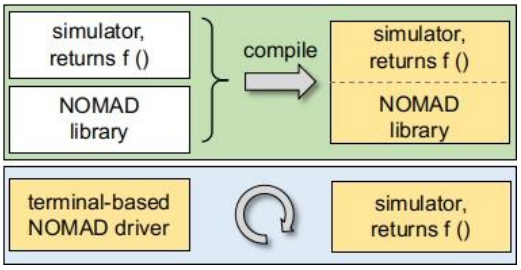

Fig. 3: Two methods to use the NOMAD tool.

The code, in the form of a library, can be linked with an objective function provided by the user; the result is a stand-alone executable that calculates the objective function and runs the optimization. This form can be exploited using compiled languages, but there are interfaces for Matlab and Python. Another alternative is to have the objective function implemented in a separate application, that will be called by a sort of NOMAD driver that sends command line parameters and collects output from stdout [29].

In the case of SLEUTH, the first method requires incorporating the library into the simulator, and the second is not compatible with the high latency of its start-up code. In this study, a third alternative was devised. We modified the simulator to receive parameters using Unix domain sockets [30]. A small application embedding the NOMAD library was implemented, replacing the terminal-based driver, as shown in Fig. 4.

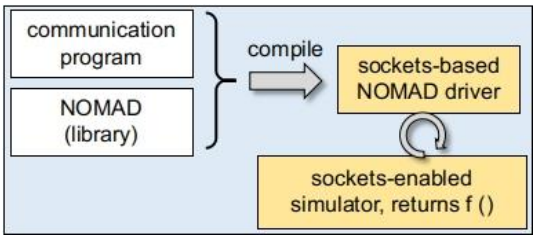

Fig. 4: Our implementation of NOMAD with SLEUTH.

Other alternatives to the architecture shown in Fig. 4. include the use of file and memory sharing, and named pipes. In the case of file and memory sharing it would be necessary to implement a synchronization scheme [31], signaling the moment when each of the endpoints - SLEUTH and NOMAD - would have finished computing metrics and parameters, respectively. Named pipes implement synchronization in a transparent way, and could be used in place of Unix domain sockets.

2.3 ANALYSIS CRITERIA

Calibration of SLEUTH and of many other cellular automata models of urban growth is a difficult process [10,12-16]. The key issue is to extract a maximum of

information from a minimum number of simulation runs. Optimization must aim at reducing the number of tested configurations, but at the same time, it must also guarantee a certain level of quality of results.

The performance of optimization algorithms is evaluated in terms of convergence rates, which, in the case of non-gradient methods, relates to decreasing lengths of the search step [18,32]. Usually, the results are represented on graphs showing target values as a function of the number of iterations, a concept similar to the technique of time-to-target plot [33].

Generally, it is preferable to use a large set of points to perform this type of comparison. In the present case, however, the simulation of different regions requires geographical data, and obtaining that information and preparing the files is a lengthy task *per se* [34]. In addition to that, long processing times limit the number of data points available for analysis [24,25].

Performance profiles are an instrument to compare the relative efficiency of optimization algorithms [35]. The method takes the best value found by the algorithms as a reference, and then computes a distribution function of the results. Plots of distributions are a means to depict the relative performance of optimizers. When the evaluation of the cost function is too expensive, a technique developed later, data profiles, is a better alternative [36]. The main difference between the two is that performance profiles compute a ratio based on the number of problems solved within a given threshold, while data profiles are calculated with respect to the number of function evaluations.

In this paper data profiles were used, with an adjustment that corresponds to selecting the full range of values to compute a distribution [36]. We begin by recording the whole set of trial points $X^k = \{\boldsymbol{x}^k_1 \ldots \boldsymbol{x}^k_{n(k)}\}$ and respective function values for each optimization algorithm k=1,... and also recording the best result $r^* = min\ \{\ f(\boldsymbol{x})\ \}$, among all tests. Then we calculate:

$$d^k(\alpha) = |\boldsymbol{x}^k : \frac{f(\boldsymbol{x}^k)}{r^*} < \alpha| \quad (1)$$

, where the symbol | stands for cardinality of a set and α varies between 0 and 1. The curve d^k indicates the number of times a method *k* produces results which are at least α percent as good as r*. By plotting d^k we get a visual description of the relative efficiency of an optimization method to explore the search space.

Here, the objective function *f* was the same metric implemented in the genetic version of SLEUTH: OSM. This choice does not mean an endorsement of this metric for calibration of the model, but analyzing this matter is not part of the scope of this study.

2.4 SIMULATION SETUP AND DATA COLLECTION

Preparing layers of geographical data for SLEUTH is a demanding task, and this kind of data is not readily publicly available. In this work, two datasets, D1 and D2, were employed with the three optimization methods: brute force; genetic algorithm; and NOMAD.

The first set of data layers, D1, were the same used in [20]. All images had 1242 x 1339 pixels, corresponding to a scale of 30 meters. The data layers included:

- a slope layer, representing in gray scale the steepness of terrain as a percentage;
- an exclusion layer, black and white, identifying areas where urbanization is not allowed;
- a set of urban footprints, black and white, in intervals of 3 years, between 1984 and 2017.
- road maps, in gray scale, for the years 1984, 1996 and 2017;

Sample images are shown in Fig. 4. At the left, an urban footprint; white pixels represent urban areas. At the right of Fig. 4, a road map; the brighter the pixel, the greater the importance of the road.

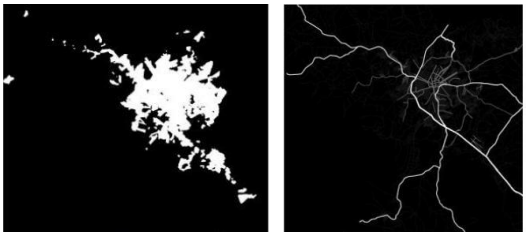

Fig. 4: Example of SLEUTH input layers.

The second dataset, D2, was composed of the images used in [37]. It includes layers for slope, exclusion, and hillshade; and images in 5-years intervals, from 2000 to 2015, depicting roads and urban footprints. Images were square, with 1116 pixels of edge and corresponding to approximately 71.5 meters per pixel.

SLEUTH supports two types of simulation: a binary-mode where the state of each cell is either urban or not; and a category-mode with different classes of land use. We chose the binary-mode simulation.

In the brute-force approach, we opted for not following the long procedure suggested in the documentation of SLEUTH, and shown in Fig 1. It was decided to make a single pass using images of full resolution; the configuration file was set to scan the search space through all the range 0 to 100 in increments of 25. This produced a total of $5^5 = 3125$ iterations, each of which was internally repeated 2 times by the Monte-Carlo process. In a classic application of Monte-Carlo this would be a small value,

but the regularity of the simulator output, depicted in Fig. 2, indicated that noise levels were not pronounced.

For the genetic version of SLEUTH, the default parameters of the tool were used; following the instructions found in the documentation, the population was set to 55 individuals, the mutation rate was 13%, and the number of generations was set to 100.

Finally, for the optimization using NOMAD, the programming interface exposes several parameters that can be adjusted by means of special function calls. However, the user guide makes recommendations only in response to difficulties with the optimization process. This way, the various parameters of the library were also left with their default values.

III. RESULTS

The simulation scenarios presented contrasting characteristics. Calibration of dataset D1 was harder to achieve, with lower values for OSM and also for Lee-Sallee. A possible cause might be the fine temporal resolution of 3 years between images [17,20]. The second dataset required less cycles and had higher values for the metrics. Table 1 summarizes the main results.

Table.1: Summary of optimization results.

Set	Method	Cycles	Best	Point @ cycle
D1	B.F.	3165	0.00455	[1 50 1 100 1] @ **270**
	G.A.	8024	**0.00472**	[8 99 95 100 91]@7544
	NOMAD	7999	0.00465	[8 96 73 99 41] @ 2818
D2	B.F.	3165	**0.68370**	[100 75 50 1 1] @ 2925
	G.A.	929	0.66965	[66 66 57 1 48] @ 914
	NOMAD	780	0.64048	[21 100 90 1 1] @ **150**

The worst value for both datasets was 0. The best result r* for dataset D1 was found at iteration 7544 by the GA optimization. The Brute-Force method found its best value at iteration 270, but this is not a fast result since, by design, the algorithm blindly scans the whole search space. NOMAD came in second place with less than half the effort of GA to reach 98.5% of r*.

For the second dataset, NOMAD was by far the fastest algorithm; the best value for OSM was found after 150 iterations only, although the method executed additional cycles to ascertain that no further improvement was possible. On the other hand, NOMAD also had the lowest global result, behind G.A. in second place and Brute-Force in the first position.

The G.A. implemented in SLEUTH selects initial points along of a diagonal that traverses the search grid, with coordinates (0+Δ, 0+Δ, 0+Δ, 0+Δ, 0+Δ) for increasing values of Δ. NOMAD utilises a variation of a simplex-based algorithm, and is likely to be more sensitive to the choice of the starting point. NOMAD was tested with points (0, 0, 0, 0, 0) and (50, 50, 50, 50, 50), but the second choice caused the algorithm to obtain worst results. A plot of the optimization trajectories provides an intuitive comparison of G.A. and NOMAD. This is shown in Fig. 5, for the dataset D1 (with similar characteristics found for D2).

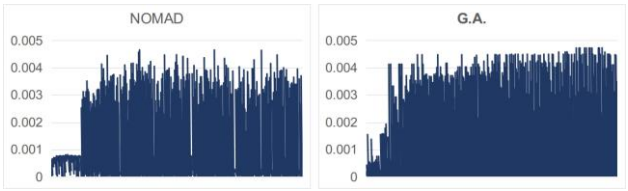

Fig. 5: Optimization history of NOMAD and G.A.

Fig. 5 shows that NOMAD succeeds in escaping a region of low values and, subsequently, it searches for areas with high values. However, sporadically it touches points of bad quality. The algorithm G.A. implemented in SLEUTH, by comparison, seems to be more elitist and avoids low points altogether.

It is interesting to note that the graphs on Fig. 5 were created from simulator logs and show all the points evaluated. This way, they offer a general view of the heuristics followed by each algorithm.

The concept of data profiles allows us to draw a more detailed comparison. The question to address is the relative efficiency of the algorithms to yield results, instead of comparing only peak values, or number of function evaluations. This is more relevant in the present context, because of low parameter sensitivity (which leads to slow progression), and the difficulty to find a global maximum.

The Fig. 6 shows a data profile graph for dataset D1. We use the same perspective as [31] and plot $d^{-1}(\alpha)$; this way the data profiles illustrate the relative computation effort to attain a minimum αr^*.

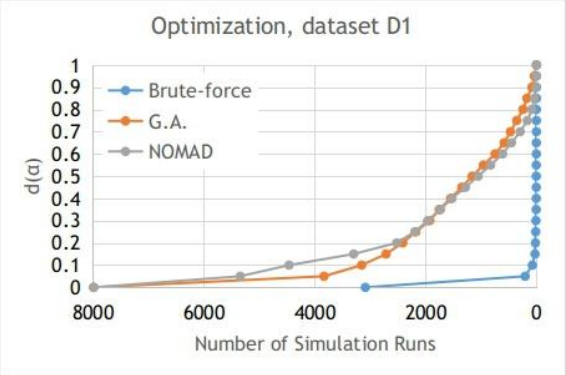

Fig. 6: Plot of $d^{-1}(\alpha)$ for dataset D1.

To exemplify how to interpret the plot in Fig 6, let's chose α=0.3. The graph shows that approximately 2000 function evaluations of NOMAD (1964 to be exact) produced values equal or better to 30% of the best result r*= 0.00472 . The genetic algorithm version of SLEUTH comes close, with 1927 evaluations. By comparison, in the Brute-Force approach, only 11 function evaluations attained the same mark. Moreover, along the interval $0.2 < \alpha < 0.8$, the exhaustive search shows a low probability of finding an adequate set of parameters. Fig. 6 also shows that, for values of α above 0.5, G.A. obtained slightly better results than NOMAD.

The results for dataset D2 are shown in Fig 7.

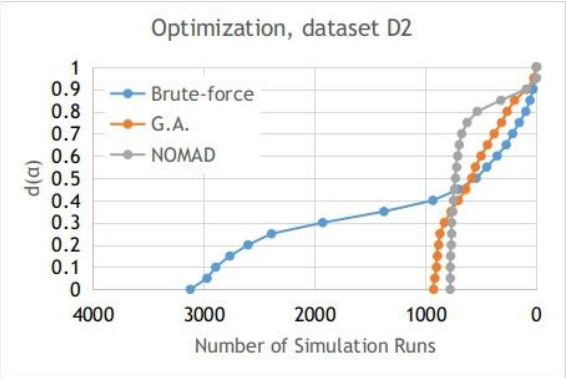

Fig. 7: Plot of d(α) for dataset D2.

Once again, NOMAD and G.A. obtained significant savings in the number of function evaluations, orienting the search towards regions of good potential. In this test NOMAD jumped ahead of G.A. for values $\alpha \geq 40\%$. A main difference between NOMAD and G.A. comes from the fact that the first follows a trajectory while exploring the space, whereas the latter is more flexible and can take random jumps. In theory, G.A. has a greater chance of finding a global extreme, while NOMAD has the potential to converge faster; this was the case with dataset D2.

As a last verification, it was evaluated the relative distance between solutions found by each optimization algorithm. As it was mentioned, the five parameters control the results of the simulation and are utilised to characterize and compare trends of city growth [20]. If the calibration returns points that are too far from each other, this might be a sign that local solutions were found.

Table.2: Distance between the 20 best solutions.

	radius of hyperball	
Dataset	**NOMAD**	**G.A.**
D1	20.3004	45.8189
D2	2.9580	45.3927

The values in Table 2 correspond to the radius of a hyperball containing the 20 best 5-dimensional points produced by each optimization algorithm. The radius was determined using the algorithm described in [38]. It can be seen that NOMAD tested more points around the same neighborhood, in comparison to G.A. This is another indication that, comparatively, if NOMAD might converge faster, G.A. might cover a more extensive area.

IV. CONCLUSION

The outputs of a computer simulation can be treated in certain contexts as functions of input parameters. An example of this is the calibration of models of urban growth, where a function indicates the quality of the simulated results. In the case of SLEUTH, characteristics as nondeterminism, discontinuities, and nonlinearity make it harder to optimize parameters. In addition to these aspects, the execution of the model has a high computational cost. This negatively impacts studies that aim to contrast scenarios and perform what-if analysis, and justifies the interest in accelerating the calibration process.

This study used named sockets to couple SLEUTH to the NOMAD optimization library. This choice ensures fast communication, and required minimal changes in the simulator code.

The number of function evaluations in the experiments was limited in the brute force approach, if compared with the standard calibration procedure described in SLEUTH documentation. Nevertheless, the volume of data generated was sufficient to provide a baseline.

In the first dataset, genetic-algorithms and NOMAD presented close results. The two methods showed a near-linear relation between the number of function evaluations and the improvement of results in 50% of the points tested ($0.2 < \alpha < 0.7$). It must be emphasized that the analysis

refers to the relative computational effort spent to optimize the calibration, and not the convergence speed. As indicated on Fig 2., both the GA version of SLEUTH and NOMAD spend some effort trying to escape from local minima during the search.

For the second dataset, the optimization was a lot faster, and NOMAD was more efficient than GA. For instance, the library provided 675 parameter configurations within 70% of r*, the double of points found by the genetic algorithm. The G.A. version obtained the best result, but the corresponding f was only 4.5% better than NOMAD.

The SLEUTH parameters in Table 1 show a certain disagreement between methods. This is in accordance with the fact that the cost function has several minima. However, large deviations of values would conflict with the idea of using the five parameters to characterize the growth of a city. We estimated the coherence of results by computing the smallest hyper-ball holding the 20 best points found by each optimization algorithm. The results in Table 2 indicate that, while NOMAD tries to find global optima, it also refines the search around points of greater potential. The genetic version of SLEUTH exhibited a less pronounced tendency in this sense.

Overall, NOMAD has proven to be a good solution for SLEUTH calibration, and potentially better than the genetic version of that simulator. The library can handle characteristics as non-linearity, noisy and discontinuous functions of real and integer parameters, present in many simulation models. As a consequence, this study with NOMAD and the technique of data profiles can be applied in similar situations, involving calibration of urban models but also other computer simulations that include an optimization task.

REFERENCES

[1] Kennedy, M. C., & O'Hagan, A. (2001). Bayesian calibration of computer models. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 63(3), 425-464.

[2] Tscheikner-Gratl, F., Zeisl, P., Kinzel, C., Leimgruber, J., Ertl, T., Rauch, W., & Kleidorfer, M. (2016). Lost in calibration: why people still do not calibrate their models, and why they still should–a case study from urban drainage modelling. Water Science and Technology, 74(10), 2337-2348.

[3] Sargent, R. G., & Balci, O. (2017). History of verification and validation of simulation models. In 2017 Winter Simulation Conference (WSC) (pp. 292-307). IEEE.

[4] Venkatasubramanian, V., Rengaswamy, R., & Kavuri, S. N. (2003). A review of process fault detection and diagnosis: Part II: Qualitative models and search strategies. Computers & chemical engineering, 27(3), 313-326.

[5] Rocha, F. J. P. S. P. (2012). Sistemas complexos, modelação e geosimulação da evolução de padrões de uso e ocupação do solo. University of Lisboa. PhD Thesis.

[6] Tarantola, A. (2005). Inverse problem theory and methods for model parameter estimation. Society for Industrial and Applied Mathematics.

[7] Wu, F. (2002). Calibration of stochastic cellular automata: the application to rural-urban land conversions. International journal of geographical information science, 16(8), 795-818.

[8] Santé, I., García, A. M., Miranda, D., & Crecente, R. (2010). Cellular automata models for the simulation of real-world urban processes: A review and analysis. Landscape and Urban Planning, 96(2), 108-122.

[9] Clarke, K. C., & Gaydos, L. J. (1998). Loose-coupling a cellular automaton model and GIS: long-term urban growth prediction for San Francisco and Washington/Baltimore. International journal of geographical information science, 12(7), 699-714.

[10] Clarke-Lauer, M. D., & Clarke, K. C. (2011, July). Evolving simulation modeling: Calibrating SLEUTH using a genetic algorithm. In Proceedings of the 11th International Conference on GeoComputation, London, UK (Vol. 2022, pp. 20-22).

[11] Goldstein, N. C. (2004). Brains versus brawn-comparative strategies for the calibration of a cellular automata-based urban growth model. GeoDynamics, 249-272.

[12] Wu, F. (2002). Calibration of stochastic cellular automata: the application to rural-urban land conversions. International journal of geographical information science, 16(8), 795-818.

[13] Straatman, B., White, R., & Engelen, G. (2004). Towards an automatic calibration procedure for constrained cellular automata. Computers, Environment and Urban Systems, 28(1-2), 149-170.

[14] Al-Ahmadi, K., See, L., Heppenstall, A., & Hogg, J. (2009). Calibration of a fuzzy cellular automata model of urban dynamics in Saudi Arabia. Ecological complexity, 6(2), 80-101.

[15] Newland, C. P., Maier, H. R., Zecchin, A. C., Newman, J. P., & van Delden, H. (2018). Multi-objective optimisation framework for calibration of Cellular Automata land-use models. Environmental modelling & software, 100, 175-200.

[16] Roodposhti, M. S., Hewitt, R. J., & Bryan, B. A. (2020). Towards automatic calibration of neighbourhood influence in cellular automata land-use models. Computers, Environment and Urban Systems, 79, 101416.

[17] Candau, J. T. (2002). Temporal calibration sensitivity of the SLEUTH urban growth model (Doctoral dissertation, University of California, Santa Barbara).

[18] Kolda, T. G., Lewis, R. M., & Torczon, V. (2003). Optimization by direct search: New perspectives on some classical and modern methods. SIAM review, 45(3), 385-482.

[19] Le Digabel, S. (2011). Algorithm 909: NOMAD: Nonlinear optimization with the MADS algorithm. ACM Transactions on Mathematical Software (TOMS), 37(4), 1-15.

[20] Roth, E. C. W., & Koscianski, A. (2020). Improving Forecasts of Land Use with regionalized maps in the SLEUTH model. GeoFocus. Revista Internacional de Ciencia y Tecnología de la Información Geográfica, (25), 153-174.
[21] Silva, E. A., & Clarke, K. C. (2002). Calibration of the SLEUTH urban growth model for Lisbon and Porto, Portugal. Computers, environment and urban systems, 26(6), 525-552.
[22] Lee D. R. & Sallee G. T. (1970).A method of measuring shape. Geographical Review, 60(4), 555–563.
[23] Dietzel, C., & Clarke, K. C. (2007). Toward optimal calibration of the SLEUTH land use change model. Transactions in GIS, 11(1), 29-45.
[24] Audet, C. (2014). A survey on direct search methods for blackbox optimization and their applications. In Mathematics without boundaries (pp. 31-56). Springer, New York, NY.
[25] Amaran, S., Sahinidis, N. V., Sharda, B., & Bury, S. J. (2016). Simulation optimization: a review of algorithms and applications. Annals of Operations Research, 240(1), 351-380.
[26] Wright, M. H. (2010). Nelder, Mead, and the other simplex method. Documenta Mathematica, 7, 271-276.
[27] Audet, C., Le Digabel, S., & Tribes, C. (2019). The mesh adaptive direct search algorithm for granular and discrete variables. SIAM Journal on Optimization, 29(2), 1164-1189.
[28] Audet, C., & Dennis Jr, J. E. (2006). Mesh adaptive direct search algorithms for constrained optimization. SIAM Journal on optimization, 17(1), 188-217.
[29] Le Digabel, S., & Tribes, C. (2009). NOMAD User Guide: Version 3.5 (pp. 1-45). Groupe d'études et de recherche en analyse des décisions.
[30] Stevens, W. R., Rudoff, A. M., & Fenner, B. (2003). UNIX Network Programming Volume 1: The Sockets Networking API (Vol. 3). Addison-Wesley Professional.
[31] Deitel, H. M., Deitel, P. J., & Choffnes, D. R. (2003). Operating systems. Pearson.
[32] Torczon, V. (1991). On the convergence of the multidirectional search algorithm. SIAM journal on Optimization, 1(1), 123-145.
[33] Aiex, R. M., Resende, M. G., & Ribeiro, C. C. (2007). TTT plots: a perl program to create time-to-target plots. Optimization Letters, 1(4), 355-366.
[34] Roth, E. C. W. (2019). Urban growth forecast using segmented and complete maps with the SLEUTH simulator (Master's thesis, Universidade Tecnológica Federal do Paraná).
[35] Dolan, E. D., & Moré, J. J. (2002). Benchmarking optimization software with performance profiles. Mathematical programming, 91(2), 201-213.
[36] Moré, J. J., & Wild, S. M. (2009). Benchmarking derivative-free optimization algorithms. SIAM Journal on Optimization, 20(1), 172-191.
[37] Liu, Y., Li, L., Chen, L., Cheng, L., Zhou, X., Cui, Y., Li, H. & Liu, W. (2019). Urban growth simulation in different scenarios using the SLEUTH model: A case study of Hefei, East China. Plos one, 14(11), e0224998.
[38] Gärtner, B. (1999, July). Fast and robust smallest enclosing balls. In European symposium on algorithms (pp. 325-338). Springer, Berlin, Heidelberg.