Chapter 3
Evaluation of the SimpopLocal Model

Abstract The SimpopLocal model exposes 6 free parameters that cannot be set using empirical data. This chapter presents how to evaluate SimpopLocal in spite of these degrees of freedom. A first evaluation establishes whether the model has the capacity to produce acceptable dynamics. To achieve this evaluation, the quality of the simulated dynamics is made explicit using a quantitative analysis. Based on this quantitative evaluation, an automated calibration algorithm is designed using a state-of-the-art multi-objective genetic algorithm. The results show that the model is able to produce acceptable dynamics. A second evaluation exposes the contribution of each free parameter to the capacity of the model to produce these acceptable dynamics. A novel sensitivity analysis algorithm called calibration profile is then applied. The results of this analysis show that the model can be simplified by removing one superfluous mechanism and one superfluous parameter and that all the remaining mechanisms are mandatory in the model and all the remaining parameters can be better constrained by narrowing down their definition domains.

3.1 Quantitative Evaluation

As exposed in the previous chapter, the SimpopLocal model includes 6 free parameters. To evaluate this model, a design of experiment in the space of its parameters should be carried out. In order to find out if the model works as expected the space of parameters should be explored extensively. To design the exploration of SimpopLocal several aspects have to be taken into account: 1/ which stopping criterion should be used? 2/ how to measure the quality of the computed solution? 3/ how to sample the space of parameters?

3.1.1 Stopping Criterion

In order to automate the exploration and to test a large number of configurations, a stopping criterion should be established. To do so, two kinds of constraints have to be taken into account: thematic constraints and technical ones.

D. Pumain and R. Reuillon, *Urban Dynamics and Simulation Models*,
Lecture Notes in Morphogenesis, DOI 10.1007/978-3-319-46497-8_3

SimpopLocal is an attempt to identify some of the mechanisms at works during the transition from a system of small settlements before agriculture advent (80–400 inhabitants per settlement), to a system of urban settlements (up to 7000 or 10,000 inhabitants for the bigger settlement). We cannot capture this complex transition into a single realist and unique history, therefore we decide to focus on the capacity of SimpopLocal to generate plausible dynamics with carefully chosen constraints (parameters, mechanisms, initial condition). To do so, the initial size and organization of system of settlements are set using common values and knowledge taken from specialized studies on this subject. We choosed to study the growth of 100 initial settlements with population size generated using the widely used log-normal and we used the central place theory (Christaller 1933) to distribute them geographically (Archaeomedes 1998; Johnson 1977; Liu 1996; Sanders 2012). Then we simulate the urbanization of the initial agrarian system of settlements in about four thousand years (Bairoch 1985; Marcus and Sabloff 2008). Based on this empirical values, we decided that SimpopLocal should be evaluated given its first 4000 simulation steps, which matches 4000 years of evolution of the system of cities.

On the technical aspect, we choose to represent innovations as autonomous objects into SimpopLocal. Choosing an object representation easily ensures the tracking of each innovation diffusion during simulation. Even more important, the acquisition process ensures that an innovation does not already exist in this settlement before recopy (i.e. adoption by the same settlement). One way to prevent recopy of an already existing innovation consist to store into each innovation the identification number of original innovation (before any copy). The number of innovation is therefore only growing during a run. These innovations are represented as objects which consume memory and increase the computation complexity of the model. For high values of the parameter `pCreation` this number of innovations can get arbitrary high. It would slow down the model execution and fill the computer memory. To avoid this situation we decided to establish a technical stopping criterion, by stopping the simulation when the number of innovation reaches 10,000 innovations. With this limit the model runs in approximately 1 second per simulation. This stopping criterion is purely technical and we seek to produce acceptable dynamics despite this computational limitation.

3.1.2 Expectations

Now that stopping criterion have been established, we can define a design of experiments to explore the parameter space of SimpopLocal. The widely used full factorial design of experiment is unpractical in our case. Indeed, using 10 levels for each of the 6 free parameters of SimpopLocal would produce 1 million parameter sets to evaluate. We will see bellow that this quantity of computation is affordable using modern distributed computing architecture, however checking 1 million dynamics visually is impossible.

The evaluation should be automated. The exploration of the space of parameters should directly produce a small set of parameter values which produce "expected" dynamics. We should first quantify what is an expected dynamic. To do so, we design three objectives to evaluate a single run of SimpopLocal (the lower the objective the better the dynamic):

- The objective of distribution, which quantifies the ability of the model to produce settlement size distributions that fit a log-normal distribution. To compute this objective, we evaluate the outcome of each simulations using a 2-sample Kolmogorov–Smirnov test (the deviation between the simulated distribution and a theoretical log-normal distribution having the same mean and standard deviation). Two criteria are reported, with value 1 if the test is rejected and 0 otherwise: the likelihood of the distribution (the test returns 0 if p-value >5%) and the distance between the two distributions (the test returns 0 if D-value <D[1]). In order to summarize those tests in a single quantified evaluation, we add the results of the two tests (the result of the test may be 0, 1 or 2 depending on the fit of the settlement size distribution to a log-normal).
- The objective of population, which quantifies the ability of the model to generate large settlements. The outcome of one simulation is tested by computing the deviation between the size of the largest settlement and the expected value of 10,000 inhabitants:
  $|(\textit{population of largest settlement} - 10{,}000)/10{,}000|$.
- The objective of simulation duration, which quantifies the ability of the model to generate expected configurations in a suitable length of time (in simulation steps). The duration of one simulation is tested by computing the deviation between the number of iterations of the simulation and the expected value of 4000 simulation steps:
  $|((\textit{simulation duration} - 4000)/4000|$.

3.1.3 Handling the Stochasticity

SimpopLocal is a stochastic model, meaning that its outputs are probability distributions. To estimate the quality of the dynamics produced by a stochastic model for a given set of parameters, the model should be run several times or replicated using independent random number streams. The results of the replications are independent realizsations of the output random variates of the model. The quantitative expectations for this model should then be expressed as descriptive statistics on the output distributions.

In our case we want to find suitable dynamics which are robust to stochasticity. It means that we seek parameter values such as the model dynamics gets as close as possible to the three previously defined objectives as often (for as many realizations

[1]Computed with $\alpha = 1.36$.

of the dynamics) as possible. To take into account the stochasticity of the model we define the three new evaluation objectives as follows:

- the aggregated distribution objective: the mean of the distribution objective among the replications,
- the aggregated population objective: the median of the population objective among the replications,
- the aggregated duration objective: the median of the duration objective among the replications

Note that the scale of these objectives are independent from the number of replications. It means that an evaluation based on *n* replications can be quantitatively compared with another based on *m* replication. This property will be useful for the following of this chapter.

3.2 Automated Calibration

3.2.1 Optimization Heuristic

Now that we have defined a quantitative evaluation of the model dynamics, we can sample the input parameter space to test if the model is able to produce suitable dynamics. Several methods are available to sample the space of parameters. They can can be split in two categories:

- the a priori samplings methods sample the space of parameters once and for all and then evaluate the model for each of the sampled points. In this category, the regular lattice is often used by modellers. Other samplings, with better space coverage are available such as the Latin Hypercube Sampling[2] or the Sobol Sequence[3] (for a full review on parameter space sampling report to Kleijnen 2007). These methods are simple to carry on and often allow rigorous statistical analysis of the results. However, they might be inefficient at finding acceptable dynamics when very few knowledge is available on the possible range of the input parameters.
- the iterative samplings take into account the already computed evaluations in order to generate more samples. This category contains instance calibration processes based on optimization algorithms (Stonedahl 2011), approximate Bayesian computation (Beaumont 2010; Lenormand et al. 2012), Calibration Profiles (Reuillon et al. 2015), Pattern Search Exploration (Chérel et al. 2015).

For SimpopLocal we have chosen to calibrate it through an iterative process based on a genetic algorithm. Since we have 3 objectives we used the well-established NSGA2 multi-objective optimization algorithm (Deb et al. 2000) using the 6 free

[2]https://en.wikipedia.org/wiki/Latin_hypercube_sampling.

[3]https://en.wikipedia.org/wiki/Sobol_sequence.

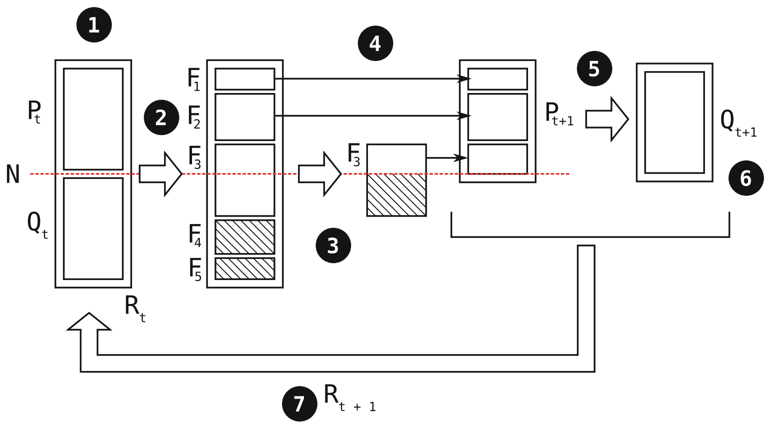

Fig. 3.1 NSGA 2 step by step procedure, inspirated by original schema from Deb et al. (2000)

parameters of SimpopLocal as the genome of the algorithm and the 3 objectives as the multi-objective fitness of the algorithm.

As represented on Fig. 3.1, the NSGA-2 algorithm computes the evolution of a population of size $2 * N$ called R. At the first iteration of NSGA-2 (I_0), the algorithm is initialized a population (P_0) of randomly generated solutions. Starting from I_1 the algorithm loops until convergence, performing the following steps:

- In step 2, the population (R_t) is ranked using **non-dominated sorted (NDS) algorithm**.[4] This algorithm uses the Pareto dominance to compute so-called fronts (a definition of Pareto dominance, and a detailed example of front computation is given later in this section). It groups individuals of the population by Pareto front $F_{1...n}$ using a multi-objective fitness. The individuals that belongs to front F_1 are dominated by no individual in the populations R_t. The individuals of front F_2 are dominated by no individual in the population $R_T - F_1$, the population where the individuals of F_1 are excluded ... and so forth.
- In step 3, the NSGA-2 algorithm computes a new population from R_T by selecting the best individual of R_T. The algorithm first adds all the individual of F_1, then the ones of F_2, the ones of F_3 ... It stops just before when adding an additional front in the population make it bigger than N individuals.
- In step 4, the algorithm adds a sub-part of the next front in order to complete the new population (it should reach a size of exactly N individuals). The individuals of a front cannot be discriminated by their objective values (by definition they constitute compromise solutions), therefore NSGA-2 uses another ranking based

[4]Invented by Goldberg (1989) but first implemented by Deb in NSGA (Deb et al. 2000).

on a diversity metric called the crowding distance operator. This selection based on a diversity metric helps maintaining a diversity of solutions in the population and not to converge too early in a local minimum. This population of size N is called P_{t+1}.

- In step 5, a set N parameter values (or genomes) is generated is generated by recombining and mutating individuals taken at random from P_{t+1}. This set is called Q_{t+1}.
- In step 6, this new offspring population Q_{t+1} is evaluated by running the fitness function. Each individual gain a new vector of value which contain evaluation for each objective function.
- In step 7, the new Q_{t+1} and already existing population P_{t+1} are merged into population R_{t+1}. Some convergence criterion is then tested. If the convergence has not been reached, the algorithm go to steps 2 otherwise it stops and returns R_{t+1}.

In this type of algorithm the best individuals are preserved in the R_t population used for fitness evaluation. This property is called **elitism** in evolutionary algorithms literature.

NSGA-2 heavily relies on the computation of the successive Pareto fronts. A Pareto front captures a group of individuals who are **non-dominated** by other individuals in a population. Classic definition of dominance say that "an element x_1 dominates (is preferred to) an element $x_2 (x_1 \dashv x_2)$ if x_1 is better than x_2 in at least one objective function and not worse with respect to all other objectives" (Weise 2011).

For instance in the Table 3.1, if we consider that best individuals are individuals which minimize value on objective function f_1 and f_2, we can see that I_d is better than I_c on f_1, but I_c is better than I_d on f_2. Therefore they are compromise solutions: none of them dominates the other. The set of individuals which are not dominated by any other individual of the population constitutes the Pareto front of the population. In this example, the Pareto front (which contain all non-dominated individuals) is $\{e, d, c, b, a\}$.

At step 4 of the algorithm, the Non-Dominated Sorting algorithm (NDS) computes successive fronts $F_{i...n}$ by iteratively removing non- dominated individuals $\{I_\varnothing\}$ from population P. In the example of Fig. 3.2, the front F_2 is computed by removing all non-dominated individuals (i.e. individuals in F_1) from population and then computing the Pareto front of this population. In this example we remove $\{e, d, c, b, a\}$, so the new Pareto front of P is equal to $\{f, h, j, k, l\}$.

3.2.2 Adaptation of NSGA2 to a Stochastic Model

A problem when using genetic algorithms to calibrate simulation models is that some of them do not cope well with stochasticity. This is especially the case for algorithms of type $\mu + \lambda$ (such as NSGA2), which preserve best solutions between

Table 3.1 Example of fitness value computed using NDS fitness algorithm and a population of individuals evaluated on two objective function f_1, f_2

Individuals	f_1	f_2	Dominated by	Fitness value
a	3.5	1	∅	1
b	3	1.5	∅	1
c	2	2	∅	1
d	1	3	∅	1
e	0.5	4	∅	1
f	0.5	4.5	{e}	2
g	1.5	4.5	{d, e, f, h}	3
h	1.5	3.5	{d}	2
i	2	3.5	{c, d, h}	3
j	2.5	3	{c, d}	2
k	3.5	2	{a, b, c}	2
l	4.5	1	{a}	2
m	4.5	2.5	{a, b, c, k, l}	3
n	4	4	{a, b, c, d, e, h, i, j, k, o}	5
o	3	4	{b, c, d, e, h, i, j}	4

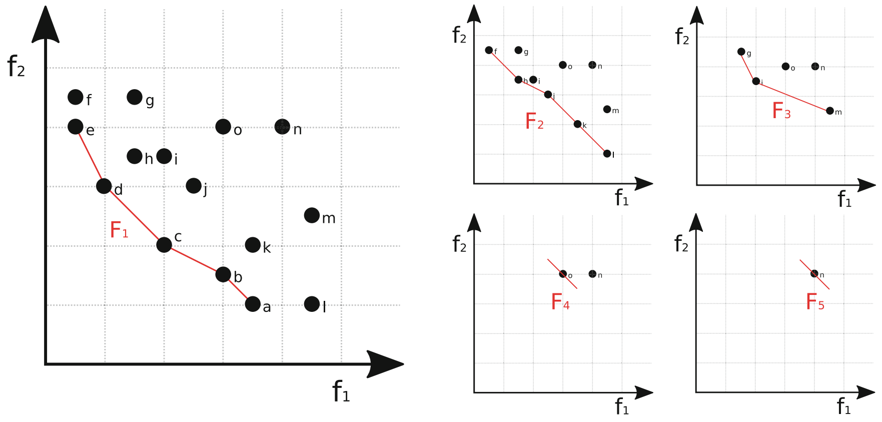

Fig. 3.2 Building steps example for NDS algorithm front computation with two objective function optimization. Algorithm produces five fronts, F_1 to F_5

the generations. In that kind of optimization the value of a solution is only estimated and not computed exactly. They can therefore be overvalued or undervalued (the quality of a solution is estimated with a significantly greater or a lower value than the one that would have been estimated given an infinite number of replications). Undervalued solutions are not very problematic for $\mu + \lambda$ genetic algorithms, they might be discarded instead of being kept, but the algorithm has a chance to retry a very similar solution later on. Conversely, the overvalued solution are very problematic for genetic algorithms, since the genetic algorithm might keep overvalued solution in the population of good solutions (because they have falsely been evaluated as good solutions) and generates new offspring solutions from them. This behaviour can greatly slow down the convergence of the calibration algorithm and even make it converge toward set of parameters producing very unstable output dynamics which are very likely to produce false positive good solutions.

To reduce the influence of the fitness fluctuation, the most commonly used approach is called "resampling". It consists in running several replications for each fitness evaluation. The computed quality for a set of parameters is then an estimation given a finite number of replications of the fitness computation. However, to limit the computation time taken to evaluate the quality of a single set of parameters during the calibration process, the number of replications is generally limited to a level which constitutes a compromise between the computation time taken to evaluate one set of parameters and an acceptable level of noise for the quality. Any number of replications, even very high, still implies that some solutions are overvalued with a non-negligible probability given that the fitness function is evaluated millions of times.

Other methods have been developed to optimize stochastic functions using genetic algorithms. Some of them are based on using the history of the genetic algorithm to estimate the probability distribution of the fitness (Sano and Kita 2002), others are based on the differences between the parents and the offspring (Tanooka et al. 1999) and others propose to use a partial order based on statistical tests (Rudolph 2001) ... Even if these methods seem statistically sound they complicate significantly the optimization algorithm, they are often based on some assumptions that are hard or impossible to verify (such as the invariance of the noise distribution over the fitness space) and they add parameters to the algorithm that are difficult to tune.

To overcome these limitations we have developed an auto-adaptive strategy to handle stochastic fitness functions in NSGA2. It is loosely related to the idea of resampling, for which only the best solutions are more precisely evaluated (presented in Branke 1998). In our method, called "stochastic resampling" we propose to evaluate the individual with only 1 replication and then to resample the individuals of the population with a fixed probability at each generation of the evolutionary algorithm. For instance, at each generation 90% of the individual offspring genomes are new genomes and 10% of the offspring genomes are already evaluated genomes randomly taken in the current population for which the algorithm computes one additional replication. The replications of each individual are stored in a vector of replications. The fitness of an individual is computed using (for instance) the median of each objective stored in the replication vector. The intuition is that in $\mu + \lambda$

genetic algorithms, best individuals survive several generations and therefore are the most likely to be resampled given that each individual has a fixed chance of being resampled at each generation. However, this fixed probability of resampling is not sufficient by itself to get an auto-adaptive algorithm. With this mechanism alone, well- evaluated solutions are very likely to be replaced by overvalued ones (new solution with a few "lucky" replications). To compensate this bias, we add technical objectives in NSGA2 in order to maximize the number of samples of a solution to the multi-objective optimization problem. Therefore, the number of replications is taken into account in the Pareto compromise elitism of NSGA2: solutions with many replications are kept even if some solutions are better on the other objectives but have been evaluated with less replications. By doing so, we let the multi-objective optimization algorithm handle the compromise between the quality of the solutions and their robustness. This method adds only two new parameters: 1/ the probability of resampling an individual at each generation 2/ the max number of samples for an individual to limit the memory used to store an individual. We propose to store the sample in a FIFO with a fixed size, therefore new samples are always taken into account even if the maximum number of replications has been reached for a given individual. This method has been implemented in the library for evolutionary computing: MGO[5] and has not been published yet.

3.2.3 Experimental Setup

To carry on the huge computation load required by the calibration of a stochastic multi-agent model using a genetic algorithm, we distributed it on the EGI,[6] a word-wide computation grid. To do so we used the framework OpenMOLE for distributed numerical experiments on simulation models[7] (this framework is described in more detail in the Chap. 6).

A classical way to distribute genetic algorithm is the technique known as the 'island model' (Belding 1995). The classical island model consists of instantiating permanent islands (isolated instances of an evolutionary algorithm) on many computers and organizing the migration of solutions between those islands. The EGI grid is a worldwide batch system on which organizing direct communications between islands running on multiple execution nodes is very challenging. Thus, we adapted the classical island model proposed in OpenMOLE to still benefit from the EGI architecture.

[5]https://github.com/openmole/mgo.

[6]http://www.egi.eu.

[7]http://www.openmole.org.

In this adapted version of the island model, a central population of 200 solutions is maintained on a central computer that orchestrates the submission of the computing jobs on the grid. Each job computes the evolution of the population of an island, which is an independent instance of NSGA2 started on a snapshot of the central population of 200 individuals at the time of submission. The 'island job' life cycle is managed by the EGI. Each job is submitted to the EGI and starts running when a slot becomes available on one of the data centres aggregated by the grid. When it starts running it is configured to run for 15 min. Using this distribution scheme, 1000 concurrent jobs are maintained (submitted + running) on the grid at any time. The OpenMOLE script for this experiment is exposed here.[8]

We executed 20,000 thousand islands of 15 min on the grid, after which we observed that the genetic algorithm is converged. An evolutionary algorithm is declared 'converged' when it makes no further improvements in the search for good solutions. One of the best metrics for measuring the convergence of the multi-objective optimization algorithm that is currently available is the stagnation of the hypervolume. The hypervolume measures the volume of the dominated portion of the objective space and its stagnation indicates that the algorithm has converged. To test if it is the case for our calibration we used the library MGO[9] to compute the evolution of the hypervolume. We considered only the solutions that are robust enough (estimated by the stochastic resampling strategy of the genetic algorithm with the maximum number of replications: 100 replications) and we used reference points (nadir) with the coordinates: distribution = 2.0, population = 2.0, simulation duration = 2.0 (the script to compute the hypervolume is available online[10]). Figure 3.3 shows the evolution hypervolume of the Pareto with the number of executed islands. It stagnates after 7000 islands have been executed.

3.2.4 Results

At the end of the evolution we get a file containing: 200 parameter values, the value of the three objectives for each of this points and the number of the samples (or replications) which have been taken into account in the computation of the objective values. In the stochastic resampling strategy candidate solutions are first evaluated with few replications and then promising solutions are resampled (evaluated with more replications), therefore in the resulting file not all solutions have been evaluated with the maximum number of 100 replications. We decide to consider only the most robust solutions in our result analysis (119 solutions among the 200 solutions proposed by the algorithm have been evaluated based on 100 replications). Among these robust solutions, 27 of them produce low (>0.1) objective values for the each of

[8]https://github.com/Geographie-cites/spinger-simpoplocal.

[9]https://github.com/openmole/mgo.

[10]https://github.com/Geographie-cites/springer-simpoplocal.

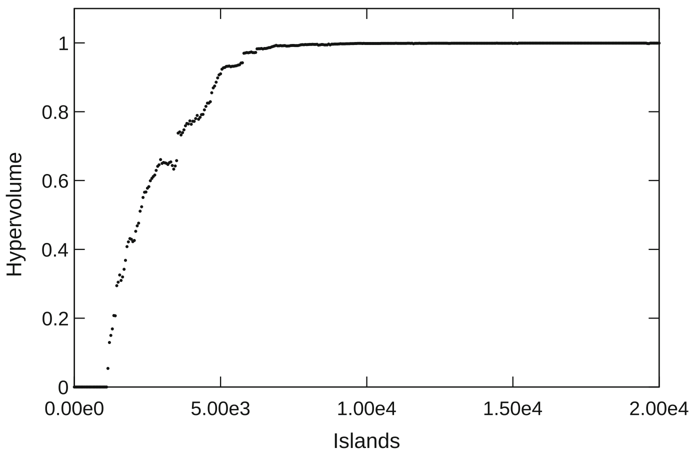

Fig. 3.3 Hypervolume of the Pareto front

the 3 objectives. The fact that the 3 objectives can take be fulfilled altogether means that these objectives are not mutually exclusive (they are compatible).

The figure exposes a run of the model for the set of parameters: *rMax* = 10134.5564655276, *innovationImpact* = 0.0100011820347467, *distanceDecay* = 0.937264929710603, *pCreation* = 0.00000119999472951903, *pDiffusion* = 0.000000879838251240765, *innovationLife* = 1529 whose fitness has been evaluated to *ksValue* = 0.015, *deltaPop* = 0.0129611448, *deltaTime* = 0.002875.

For this set of parameters, the evolution corresponds to what is expected from the model: a progressive and continuous process of hierarchical organization of the settlement system (the slope of the linear fit of the rank–size distribution shifts from 0.2 to 0.9 in 4000 years for a maximum reached size of about 10,000 inhabitants).

Further analysis show that this result is quite robust to stochasticity as shown by the low variability of the recorded final state from one simulation to another exposed on Fig. 3.4.

3.3 Calibration Profiles

In the previous section we have used automatic calibration process based on multi-objective genetic algorithms (Schmitt 2014). Nevertheless, this method produces a reduced set of candidate parameter values that represent optimal trade-off with regard to several model quality criteria. The result of the calibration process is thus solely that the model *can* reproduce the data with a given precision. It does not say anything about how often parameter sets lead to realistic behaviours, and how each parameter will change the behaviour of the model. For instance, it is often interesting to know when some parameter values would prevent the system to reach a realistic behaviour, rather than only knowing a singles set of “optimal” parameter values.

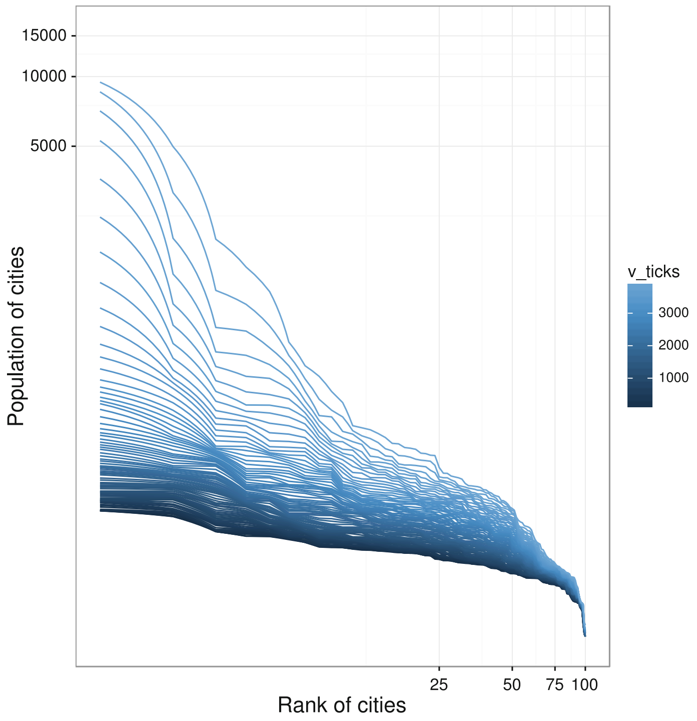

Fig. 3.4 Evolution of the rank-size distribution during a simulation of one of the best calibrated parameter settings

3.3.1 Algorithm

To compute a more global view of the parameter space, we have therefore designed a novel method that exposes the sensitivity of a single parameter on the calibration of a model independently of the other parameters (Reuillon et al. 2015). Given a function which computes a single scalar value depicting a calibration error for the model, the calibration profile algorithm computes the lowest calibration error that can possibly be obtained when the value of a given parameter is fixed and the others are free (Figs. 3.5 and 3.6). It computes this minimal error for many values of the parameter under study. The value of the parameters are sampled all along its domain of definition to produce a so-called *calibration profile*. For each sample value, the value of the remaining parameters are optimized in order to find the lowest possible calibration error. The profile can then be drawn on a 2-dimensional chart that depicts the influence of the parameter under study on the model calibration.

To produce such a profile, a naive approach would consist in executing an entire calibration algorithm for each value of the parameter under study. Current automated calibration algorithms are too computationally intensive to make this approach tractable in practice. To tackle this problem we have designed an algorithm which

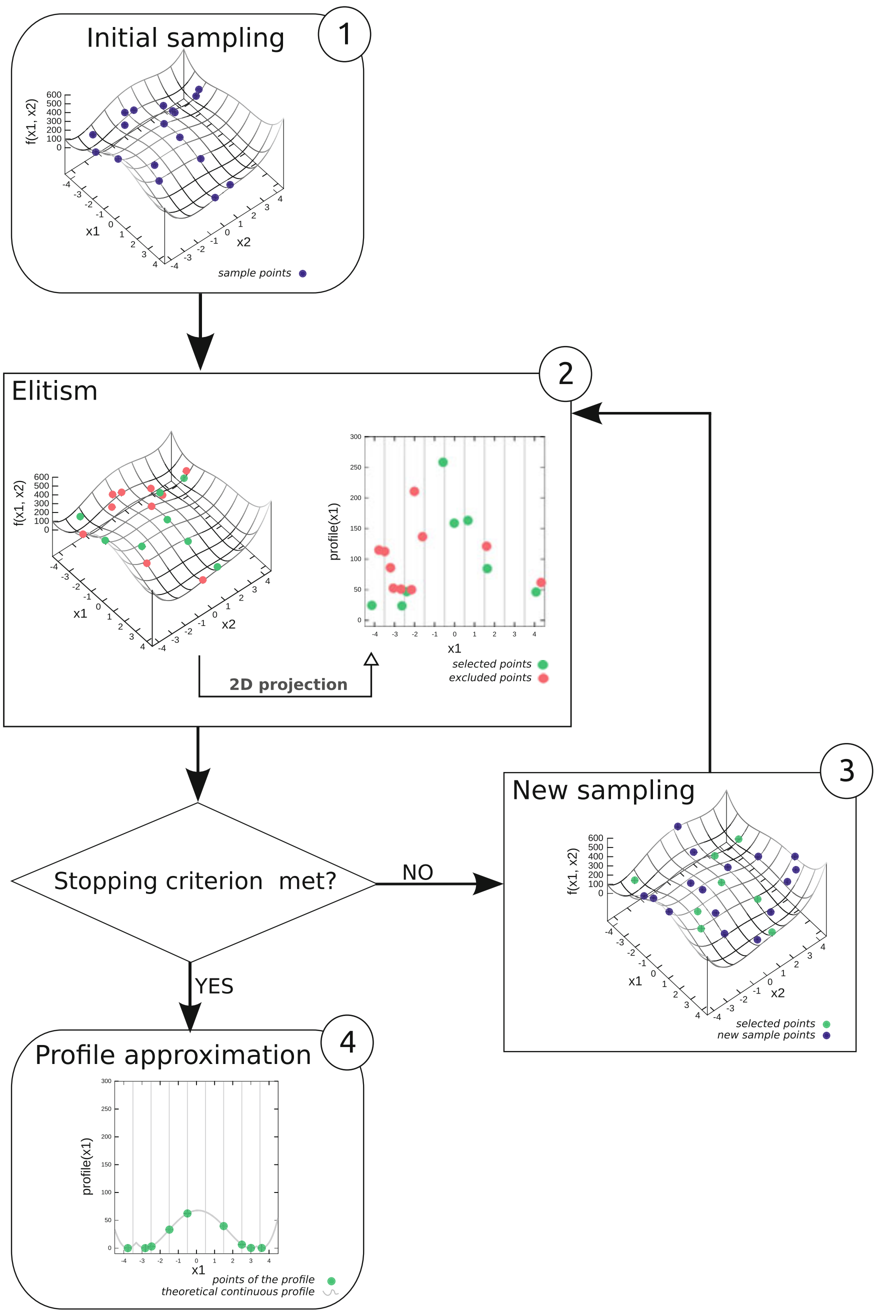

Fig. 3.5 Illustration of the calibration profile (CP) algorithm (Reuillon et al. 2015)

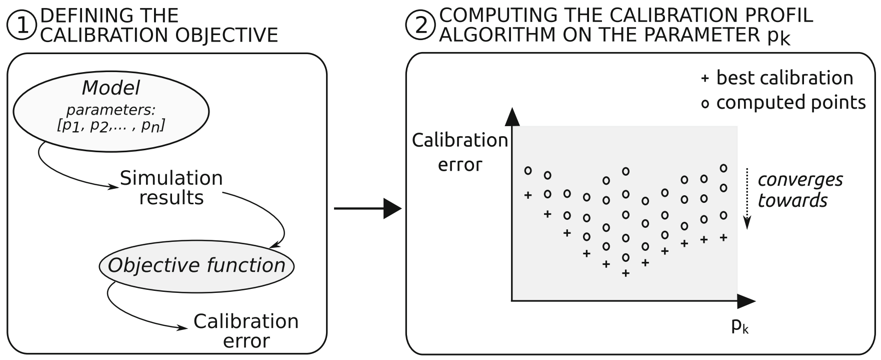
① DEFINING THE CALIBRATION OBJECTIVE
Model
parameters: [p1, p2,... , pn]
Simulation results
Objective function
Calibration error
② COMPUTING THE CALIBRATION PROFIL ALGORITHM ON THE PARAMETER pk
+ best calibration
o computed points
Calibration error
converges towards
pk

Fig. 3.6 High level representation of the calibration profile (CP) method (Reuillon et al. 2015)

computes the numerous points composing a calibration profile altogether (this algorithm has been inspired by the recently published MOLE method (Mouret 2013; Clune et al. 2013), which computes two dimensional maps of phenotype landscapes using evolutionary algorithms).

This new algorithm has been designed in the framework of evolutionary algorithms. The Fig. 3.5 illustrates the progress of this algorithm through the example of the computation of a 9-points profile along x_1 of a function $f(x_1, x_2)$. In this example, the function f represents a 2-parameters model: x_1 and x_2 and $f(x_1, x_2)$ represents the calibration error of the model. In the step 1 the algorithm randomly samples points (random values of x_1 and x_2) and computes $f(x_1, x_2)$. In the step 2, the algorithm divides the definition domain of x_1 in 9 disjoint even intervals (called niche) and keeps only the sampled point with the lowest $f(x_1, x_2)$ in each of these niches (this constitutes the elitism stage). The points that have been selected constitute a first approximation of the calibration profile of the model f along x_1. In step 3 new samples (x_1, x_2) are generated by mutating the points in the current approximation of the calibration profile and $f(x_1, x_2)$ is evaluated for each of these new points. The newly evaluated samples are merged with the existing ones and the algorithm iterates to the step 2. This iteration stops once a given stopping criterion is met after step 2. The projection of the last selected points along x_1 constitutes an approximation of the theoretical continuous profile (step 4). A detailed description of the algorithm can be found in Reuillon et al. (2015).

The calibration profile is a $\mu + \lambda$ genetic algorithm. It suffers from the same problem regarding stochasticity as the ones described in the previous section. To overcome this shortcoming, we have adapted the "stochastic resampling" strategy to this algorithm (described in the previous section). The deterministic version of CP keeps one single individual for each niche (or interval). To enable the stochastic resampling for CP, we changed this algorithm and made it keep a Pareto front in each niche (by applying the elitism strategy of NSGA2 in each niche). This Pareto front constitutes a compromise between maximizing the number of replications while minimizing the calibration error. Each Pareto front (in each niche) converges towards solutions which are both good and properly evaluated (robust to stochasticity).

3.3.2 Guide of Interpretation

A calibration profile is a 2D curve with the value of the parameter under study represented on the X-axis and the lowest possible calibration error on the Y-axis. To ease the interpretation of the profiles we propose to define an acceptance threshold on the calibration error: under this acceptance threshold the calibration error is considered sufficiently satisfying and the dynamics exposed by the model acceptable, over this acceptance threshold the calibration error is considered too high and the dynamics exposed by the model are considered unacceptable.

The computed calibration profiles may take very diverse shapes depending on the effect of the parameter of the model dynamics, however some of this shapes are recurrent. The most typical shapes are shown on the Fig. 3.7. They have been discriminated according to the variation of the values of the profile compared to the threshold value:

- The shape 1 is exposed when a parameter is restricting with respect to the calibration criterion and when the model is able produce acceptable dynamics only for a specific range of the parameter. In this case a connected validity interval can be established for the parameter.
- The shape 2 is exposed when a parameter is restricting with respect to the calibration criterion, but the validity domain of the parameter is not connected. It might mean that several qualitatively different dynamics of the model meet the calibration requirement. In this case model dynamics should be observed directly

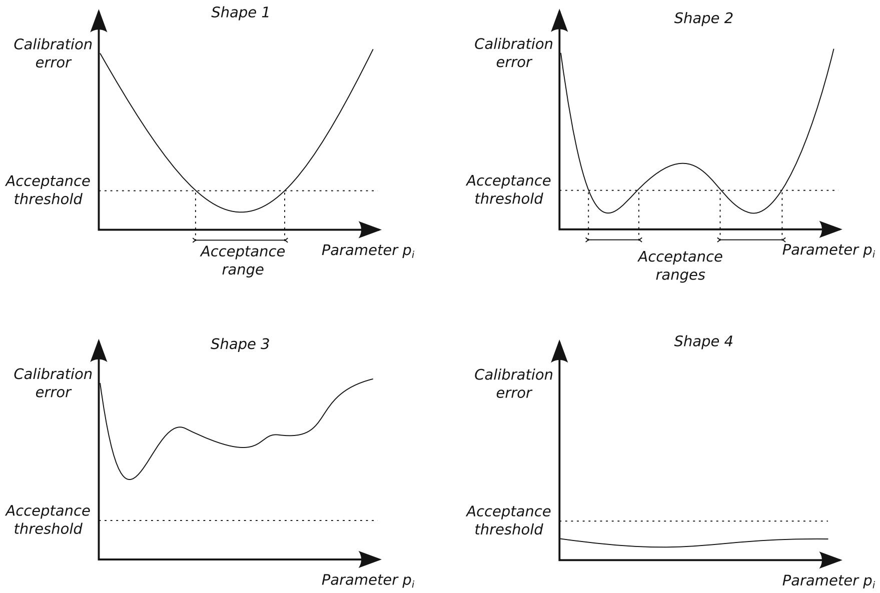

Fig. 3.7 Calibration profile (CP) algorithm (Cottineau et al. 2015)

to determine if the different kinds of dynamics are all suitable or if some of them are mistakenly accepted by the calibration objective.

- The shape 3 is exposed when the model is not possible to calibrate. The profile doesn't expose any acceptable dynamic according to the calibration criterion. In this case, the model should be improved or the calibration criterion should be adapted.
- The shape 4 is exposed when a parameter does not restrict the model dynamics with regards to the calibration criterion. The model can always be calibrated whatever the value of the parameter is. In this case this parameter constitutes a superfluous degree of liberty for the model since its effect can always be compensated by a variation on the other parameters. In general it means that this parameter should be fixed, that a mechanism of the model should be removed or that the model should be reduced by expressing the value of this parameter in function of the value of the other parameters.

3.3.3 Result Analysis

The calibration profile algorithm makes it possible to evaluate the impact of each parameter of SimpopLocal individually on the capacity of the model to produce acceptable dynamics. In Sect. 3.2, we have shown that the 3 objectives used for the calibration (distribution, duration and population) can be fulfilled altogether. To apply the calibration profiles to SimpopLocal, we consider an aggregated evaluation function f defined as the maximum value over the 3 objectives ($f = max(distribution, duration, population)$). If this value is low then a model presents acceptable dynamics. To analyse the produced results, we have established that only input parameters leading to values of f of less than 0.1 of error are considered valid. Indeed, the empirical data and theoretical knowledge that led to the definition of the objective function are not precise enough to justify a more thorough analysis of the model. This threshold is largely exceeded for some parameter values, however rendering this threshold of acceptability explicit enables the definition of credible bounds for each of the free parameter of the model. These bounds define a validity domain of each of the parameters.[11]

The Fig. 3.8 exposes the calibration profile for each of the parameter of SimpopLocal. Several interesting conclusions can be drawn by interpreting them:

- the profile for innovation life exposes that this profile has no significant impact on the capacity of the model to produce acceptable dynamics. This parameter pilots the innovation deprecation mechanism. When $innovationLife = 4000$ that the innovation deprecation time is longer than the simulation time. The facts that we can calibrate the model for this particular value indicates that the deprecation

[11]Note that the profile algorithm iteratively refines the computed profiles from high values toward lower ones through through an iterative process, therefore the proposed bounds are more restrictive than the exact ones.

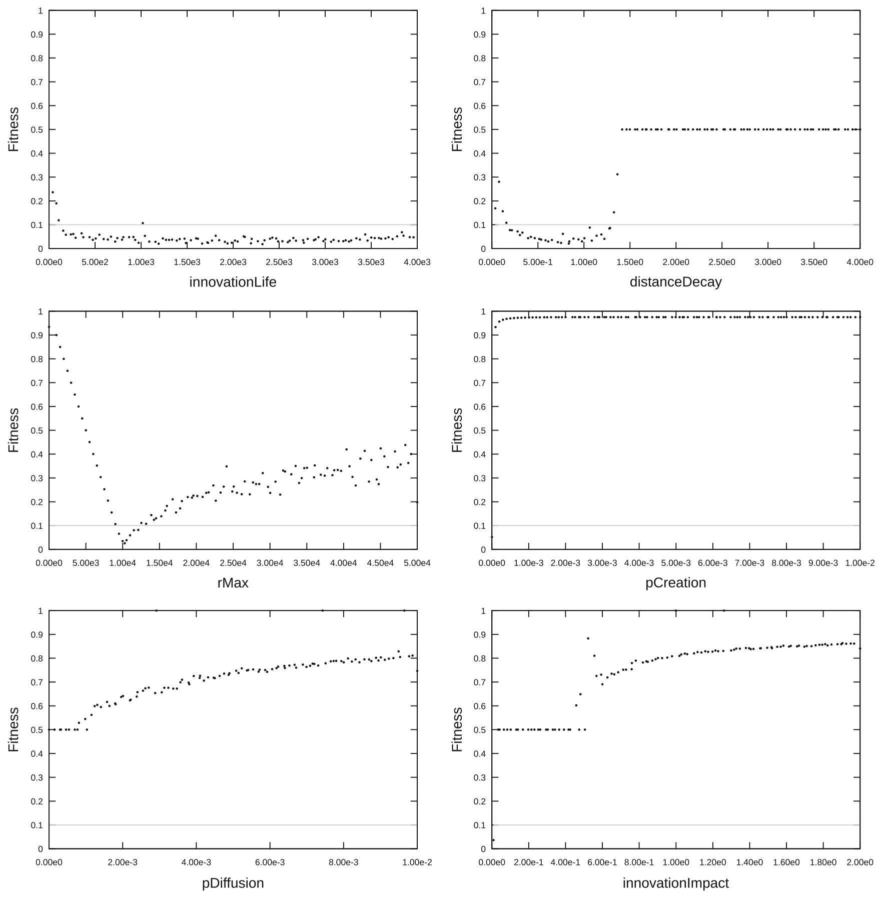

Fig. 3.8 Calibration profiles

mechanism is useless in order to reproduce acceptable dynamics and that the model can be simplified by removing it.

- The profile for *distanceDecay* exposes that the the model cannot be calibrated when the value of this parameter is lower than 0.15 or greater than 1.30. The parameter *distanceDecay* introduces a decaying effect of the distance between settlements on the diffusion of innovations from one settlement to another. When it is high, settlements are isolated from each other (and unable to exchange innovations), when it is low settlements can all exchange equally independently of their respective distances. This profiles shows that the spatial heterogeneity implemented in the model is mandatory to produce acceptable dynamics.
- The profile for *rMax* exposes that the model only produces acceptable dynamics when rMax is close to 10,000 and a best value for $rMax = 10{,}277$. The calibration objective for the size of the biggest settlement has been set to 10,000 inhabitants.

When *Rmax* is lower than 10,000 it is by construction impossible for the model to reach the 10,000 population objective. Low values of this parameter cannot achieve acceptable calibration errors. Surprisingly the calibration algorithm is not acceptable dynamics when *Rmax* is above 10,500. It indicates that this mechanism is necessary to produce acceptable dynamics and reaching settlements of a size matching empirical evidences.

- The 3 other profiles exposes only 1 or even 0 values bellow the acceptability threshold. It means that there have not been observed at the right scale (the parameter range is too broad). In the calibration experiment in Sect. 3.2, we have observed that the model exposes acceptable dynamics when they are close to 0. Therefore, we reduced the range of exploration of these 3 parameters and compute new calibration profiles.

The Fig. 3.9 exposes profiles with narrowed ranges for the 3 parameters *pCreation*, *pDiffusion* and *innovationImpact*. From this 3 curves we can deduce that:

- the validity domain of *pCreation* is between $0.4\ 10^{-6}$ and $2.2\ 10^{-6}$. For low values of *pCreation* the model cannot be calibrated. The innovations are generated too slowly to engender a sufficient growth (whatever the value of the other parameters can be). For high values of *pCreation* the system races and growth is too fast.
- the validity domain of *pDiffusion* is between $0.2\ 10^{-6}$ and $2.1\ 10^{-6}$. A very interesting aspect of this profile is that low values of pDiffusion prevent the model from producing acceptable dynamics. Noticeably when $pDiffusion = 0$ it disables entirely the diffusion mechanism of the model. At this particular point the calibration error is unsatisfying, thus this profile shows that the diffusion of innovation mechanism of the model is mandatory in order to produce realistic behaviours.
- the validity domain of *innovationImpact* is between $6\ 10^{-3}$ and $1.2\ 10^{-2}$. Under the lower bound, the impact on the settlement growth of the innovation is too low and the dynamic is too slow to reach the calibration objectives. On the contrary, when *innovationImpact* is too high the growth of the settlements is too fast to match credible dynamics (whatever the value of the other parameters can be).

3.4 Conclusion

This chapter exposes the evaluation the SimpopLocal model through a novel methodology built on top of the quantitative evaluation of the model dynamics. This methodology is generic and can be reused to other models as long as a quantitative evaluation of the model dynamics can be designed. Furthermore, all the algorithms are available in a reusable form in the free and open-source platform OpenMOLE,[12] which is presented in the Chap. 6 of this book.

The evaluation work presented in this chapter is often perceived as taking place after the modelling process, once the model is finished. On the contrary, we believe that this evaluation work should be carried all along with modelling process, from

[12] www.openmole.org.

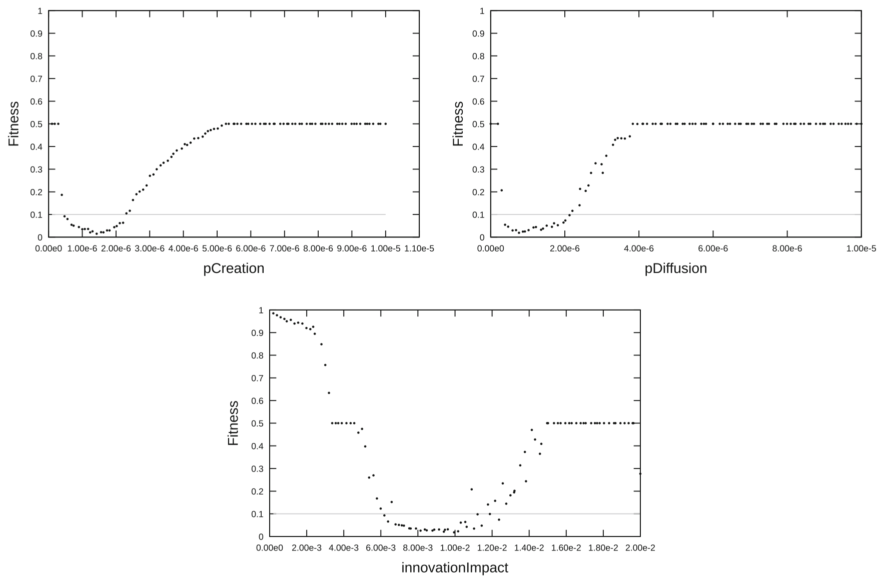

Fig. 3.9 Narrowed calibration profiles

the very early stages. From a conceptual point of view, it has the great advantage of modelling the expectation along with the model mechanisms. From a technical point of view, the quantitative evaluation can guide the modelling choices. Indeed by evaluating face-to-face candidate mechanisms, it is possible to determine which ones are the best fitted to reproduce such or such aspect of the expected dynamic. The next chapter extends this evaluation methodology and proposes an modelling framework guided by the evaluation process.

References

Archaeomedes: Des oppida aux métropoles. Anthropos, Paris (1998)

Bairoch, P.: De Jéricho à Mexico: villes et économie dans l'histoire. Gallimard, Paris (1985)

Beaumont, M.A.: Approximate Bayesian computation in evolution and ecology. Annu. Rev. Ecol. Evol. Syst. **41**(1), 379–406 (2010)

Belding, T.C.: The distributed genetic algorithm revisited. In: *Proceedings of the Sixth International Conference on Genetic Algorithms*, pp. 114–121 (1995)

Branke, J.: Creating robust solutions by means of evolutionary algorithms. Parallel Problem Solving from Nature — PPSN V: 5th International Conference Amsterdam, The Netherlands September 27–30, 1998 Proceedings, pp. 119–128. Springer, Berlin (1998)

Chérel, G., Cottineau, C., Reuillon, R.: Beyond corroboration: strengthening model validation by looking for unexpected patterns. PLoS ONE **10**(9), 1–28 (2015)

Christaller, W.: Die Zentralen Orte in Süddeutschland. Fischer, Jena (1933)

Clune, J., Mouret, J.-B., Lipson, H.: The evolutionary origins of modularity. Proc. R. Soc. Lond. B: Biol. Sci. **280**(1755), 20122863 (2013)

Cottineau, C., Chapron, P., Reuillon, R.: Growing models from the bottom up. An evaluation-based incremental modelling method (EBIMM) applied to the simulation of systems of cities. J. Artif. Soc. Soc. Simul. (JASSS), **18**(4), 9. (2015) doi:10.18564/jasss.2828. http://jasss.soc.surrey.ac.uk/18/4/9.html

Deb, K., Agrawal, S., Pratap, A., Meyarivan, T.: A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. Lect. Notes Comput. Sci. **1917**, 849–858 (2000)

Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning, p. 412. Addison-Wesley Press, Reading (1989)

Johnson, G.: Aspects of regional analysis in archaeology. Annu. Rev. Anthr. **6**, 479–508 (1977)

Kleijnen, J.: Design and Analysis of Simulation Experiments. Springer Publishing Company, Incorporated, New York (2007)

Lenormand, M., Jabot, F., Deffuant, G.: Adaptive approximate Bayesian computation for complex models (2012)

Liu, L.: Settlement patterns, chiefdom variability, and the development of early states in north China. J. Anthropol. Archaeol. **15**(3), 237–288 (1996)

Marcus, J., Sabloff, J.A.: The Ancient City: New Perspectives on Urbanism in the Old and New World. Resident Schoolar Book, vol. 2005. School for Advanced Research, Santa Fe (2008)

Mouret, J.-B.: An algorithm to create phenotype-fitness maps. In: *Proceedings of the Thirteenth International Conference on the Simulation and Synthesis of Living Systems (ALIFE 13)*, pp. 593–594 (2013)

Reuillon, R., Schmitt, C., Aldama, R.D., Mouret, J.: A new method to evaluate simulation models: the calibration profile (cp) algorithm. J. Artif. Soc. Soc. Simul. **18**(1), 12 (2015)

Rudolph, G.: A partial order approach to noisy fitness functions. In: Congress on Evolutionary Computation, Seoul, Korea, pp. 318–325. Press (2001)

Sanders, L.: Regards croisés de géographes, économistes et archéologues sur la hiérarchie des systémes de peuplement: de l'empirie aux systémes complexes. Région et Développement, **36**, (2012)

Sano, Y., Kita, H.: Optimization of noisy fitness functions by means of genetic algorithms using history of search with test of estimation. In: *Proceedings of the 2002 Congress on Evolutionary Computation, 2002. CEC'02*, vol. 1, pp. 360–365 (2002)

Schmitt, C.: Modélisation de la dynamique des systémes de peuplement: de SimpopLocal à SimpopNet. Ph.D. thesis, Université Paris 1 Panthéon-Sorbonne (2014)

Stonedahl, F.J.: Genetic algorithms for the exploration of parameter spaces in agent-based models. Ph.D. thesis, Northwestern University of Illinois (2011)

Tanooka, K., Tamaki, H., Abe, S., Kitamura, S.: A continuous age model of genetic algorithms applicable to optimization problems with uncertainties. In: IEEE International Conference on Systems, Man, and Cybernetics. IEEE SMC'99 Conference Proceedings, vol. 1, pp. 637–642 (1999)

Weise, T.: Global Optimization Algorithms - Theory and Application, 3rd edn. (2011). www.it-weise.de